DOCUMENT RESUME

ED 279 351 IR 051 875

AUTHOR Rabitti, Fausto, Ed.
TITLE ACM Conference on Research and Development in Information Retrieval: Proceedings of the Annual Conference of the Association of Computing Machinery (Pisa, Italy, September 8-10, 1986).
INSTITUTION Association of Computing Machinery, Baltimore, MD.
REPORT NO ISBN-0-89791-187-3
PUB DATE Sep 86
NOTE 295p.; Most of text is in small print. Some charts and tables may be marginally legible.
AVAILABLE FROM ACM Order Department, P.O. Box 64145, Baltimore, MD 21264 (Order No. 606860; $24.00 member, plus $4.00 handling).
PUB TYPE Collected Works - Conference Proceedings (021) -- Reports - Research/Technical (143)

EDRS PRICE MF01/PC12 Plus Postage.
DESCRIPTORS Automatic Indexing; Developed Nations; Display Systems; Expert Systems; Foreign Countries; *Information Retrieval; *Information Storage; International Organizations; Man Machine Systems; Research and Development; Search Strategies; *Systems Analysis; Trend Analysis
IDENTIFIERS Cluster Based Retrieval; Europe (West); Information Management; North America; Probabilistic Indexing

ABSTRACT

Intended to identify and encourage research, development, and applications of information retrieval, the principal objective of this conference was to provide an international forum to promote an understanding of current research and to stimulate the exchange of ideas and experiences in information retrieval systems. Introductory material for these proceedings include listings of the conference chairpersons, officers, committee members, and sponsors. The full text is then provided for two keynote speeches: "Recent Trends in Information Retrieval" (G. Salton) and "Using Structural Representation of Anomalous States of Knowledge for Choosing Document Retrieval Strategies" (N. J. Belkin and B. H. Kwasnik). The remaining 33 papers in this collection, which are also given in full text, are organized by the conference sessions at which they were presented: (1) Office Systems; (2) User Interfaces; (3) Storage Structures; (4) Linguistic Retrieval; (5) Information Retrieval Systems; (6) Clustering; (7) Retrieval Strategies; (8) Knowledge-Based Information Retrieval; (9) Learning Systems; and (10) Probabilistic Retrieval. An index is provided of the presenters, who came from Canada, Denmark, Finland, France, Ireland, Israel, Italy, Switzerland, the United Kingdom, the United States, and West Germany. (KM)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED279351

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
This document has been reproduced as received from the person or organization originating it.
Minor changes have been made to improve reproduction quality.
Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

1986 - ACM CONFERENCE ON RESEARCH AND DEVELOPMENT IN

INFORMATION RETRIEVAL

EDITED BY FAUSTO RABITTI

PALAZZO dei CONGRESSI
Via Matteotti, 1
PISA - ITALY

SEPTEMBER 8 - 10, 1986

IR051875

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
ACM
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Organization of the 1986 ACM Conference on Research and Development in Information Retrieval

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval.

Additional copies of these proceedings may be ordered prepaid from

ACM Order Department
P.O. Box 64145
Baltimore, Maryland 21264
ACM Order 606860
ISBN 0-89791-187-3

o from
ACM - CNR CONFERENCE
IEI - Via S. Maria, 46
56100 PISA, Italy

Price: member : $24
non-member: $18

Finito di stampare nel Luglio 1986 presso le Officine Grafiche
della Lito-Tipografia VIGO CURSI - Via S.Maria, 77 - Pisa.

PREFACE

The ACM Conference on Research and Development in Information Retrieval is being held this year in Pisa (Italy). This conference is traditionally held on alternate years in North America and in Europe. In 1985, it took place in Montreal (Canada) and next year it will be in New Orleans (USA). In Europe, it has already been held three times in England and once in West Germany; this is the first time in Italy.

This conference is intended to identify and encourage research, development and applications in Information Retrieval. Its principal objective is to provide an international forum to promote an understanding of current research, and to stimulate the exchange of ideas and experiences in information retrieval systems. The proceedings of the conference constitutes, year after year, an important record of the scientific and technical evolution in the area of information retrieval throughout the world.

The 1986-ACM Conference on Research and Development in Information Retrieval has been sponsored by the Italian National Research Council (CNR), a public organization which performs, coordinates and supports research activities in various fields. Luigi Rossi Bernardi, the President of CNR, is the chairman of the conference. Most of the organization has been carried out in Pisa at the "Istituto di Elaborazione dellInformazione" (IEI), an institute of CNR working in information processing. We should like to thank the Director of IEI, Franco Denoth, for his support and we are deeply grateful to all the staff of IEI who have contributed towards the organization of the conference. In particular, the efforts of Ettore Ricciardi, our treasurer, and of Manuela Mennucci, Anna Passerotti and Carol Peters have been vital to the conference.

We have benefitted from the collaboration of several important groups and organizations, such as the ACM Special Interest Group in Information Retrieval, the BCS-IRSG (British Computing Society), the ESA-IRS (European Space Agency), and, within Italy, the AICA-GLIR (the working group in I.R. of the Italian Computing Society, with the active cooperation of its chairman, Maristella Agosti) and IDI. The conference has also received a generous financial support from several companies, including IBM Italy, Siemens Data, Olivetti, Cerved, Sperry, Jackson and Logos.

We should like to express our appreciation to the members of the Conference Program Committee, and in particular to Gerard Salton, chairman the program committee for North America, who have all contributed to the selection of the papers contained in this book and which are being presented at the conference.

Fausto Rabitti

Conference Chairman

LUIGI ROSSI BERNARDI

President of CNR, Italy

Program Chairmen

FAUSTO RABITTI
IEI-CNR, Italy

GERARD SALTON
Cornell University, USA

Treasurer and Local Arrangements

ETTORE RICCIARDI, IEI-CNR, Italy

Program Committee

M. AGOSTI - Italy
P. BOLLMAN - W. Germany
Y. CHIARAMELLA - France
E. CHOURAQUI - France
F. NALDI - Italy
J.H. SCHEK - W. Germany
J. TAIT - U.K.
C.J. VAN RIJSBERGEN - U.K.
P. WILLETT - U.K.
R. ALLEN - USA
B. CROFT - USA
T. DOSZKOCS - USA
M. LESK - USA
C.T. YU - USA

Organizing Committee

F. RABITTI - IEI-CNR, Pisa
E. RICCIARDI - IEI-CNR, Pisa
M. AGOSTI - University of Padua
F. NALDI - SIAM-CNR, Milan
E. ONORATO - ESA-IRS
V. MUSSO - IDI
M. MORELLI - IBM Italia
C. MORTARINO - SIEMENS DATA
V. TOGNANA - CERVED
D. MONACO - OLIVETTI
G. CALDARA - SPERRY
F. CANAVESE - G.E. JACKSON
C. COCOLA - LOGOS

CONFERENCE SECRETARIAT

M. MANNUCCI, A. PASSEROTTI
IEI-CNR Via S. Maria, 46
56100 Pisa (Italy)
Tel. +39-50-500159 * Tlx 590305 IEICNR I

THIS CONFERENCE IS SPONSORED BY

CONSIGLIO NAZIONALE DELLE RICERCHE

IN COOPERATION WITH

ACM - SIGIR

A.I.C.A. - GLIR

Associazione Italiana per l'Informatica ed il Calcolo Automatico

BCS - IRSG

idi INFORMAZIONE DOCUMENTAZIONE INDUSTRIA

ESA - IRS

THIS CONFERENCE IS SUPPORTED BY

Istituto di
Elaborazione
dell'Informazione

IBM ITALIA

Siemens Data

SOCIETÀ NAZIONALE DI INFORMATICA
DELLE CAMERE DI COMMERCIO ITALIANE

CONFERENCE PROGRAM

monday september 8, 1986

SESSION 1: KEYNOTE SPEECH

Recent trends in automatic Information Retrieval. 1
G. SALTON (USA)

Using structural representation of anomalous states of knowledge for choosing document retrieval strategies. 11
N. J. BELKIN, B. H. KWASNIK (USA)

SESSION 2: OFFICE SYSTEMS
Chairman: F. RABITTI

Document presentation and query formulation in Muse. 23
S. GIBBS, D. TSICHRITZIS (Switzerland)

An approach to multimedia information management. 31
S. GALLELLI, C. IACOBELLI, P. MARCHISIO (Italy)

Methodological issues for the design of an office information server. 39
T. TRUCKENMÜLLER, M. RATHGEB (W. Germany)

SESSION 3: USER INTERFACES
Chairman: B. CROFT

IR, NLP, AI and UFOS: or IR-relevance, Natural Language Problems, Artful Intelligence and User-Friendly Online Systems. 49
T. DOSZKOCS (USA)

The visual display of information in an Information Retrieval environment. 58
D. CROUCH (USA)

Improved subject access, browsing and scanning mechanisms in modern on-line IR. 68
P. INGWERSEN, I. WORMELL (Denmark)

SESSION 4: STORAGE STRUCTURES
Chairman: P. WILLETT

S-Tree: A dynamic balanced signature index for office retrieval. 77
U. DEPPISCH (W. Germany)

Improved hierarchical bit-vector compression in document retrieval systems. 88
A. S. FRAENKEL, S. T. KLEIN, Y. CHOUEKA, E. SEGAL (Israel)

Text compression using prediction. 97
J. TEUHOLA, T. RAITA (Finland)

tuesday september 9, 1986

SESSION 5: LINGUISTIC RETRIEVAL
Chairman: Y. CHIARAMELLA

Incorporating syntactic information into a document retrieval strategy: An investigation. 103
A. F. SMEATON (Ireland)

CALIN: A user interface based on a simple natural language. 114
P. BOSC, M. COURANT, S. ROBIN (France)

Solving grammatical ambiguities within a surface syntactical parser for automatic indexing. 123
C. BERRUT, P. PALMER (France)

SESSION 6: INFORMATION RETRIEVAL SYSTEMS
Chairman: D. KRAFT

A design of a distributed full text retrieval system. 131
I. A. MACLEOD, P. MARTIN, B. NORDIN (Canada)

REALIST: Retrieval aids by linguistics and statistics. 138
G. THURMAIR (W. Germany)

COREL: A conceptual retrieval system. 144
M. K. DI BENIGNO, G. R. CROSS, C. G. DEBESSONET (USA).

SESSION 7: CLUSTERING
Chairman: P. BOLLMAN

Hierarchic document classification using Ward's clustering method. 149
A. EL-HAMDOUCHI, P. WILLETT (U.K.)

User-oriented document clustering: A framework for learning in Information Retrieval. 157
V. RAGHAVAN, J. S. DEOGUN (Canada)

The efficiency of inverted index and cluster searches. 164
E. VOORHEES (USA)

SESSION 8: RETRIEVAL STRATEGIES
Chairman: M. AGOSTI

On extending the vector space model for Boolean query processing. 175
S. K. M. WONG, W. ZIARCO, V. RAGHAVAN, P. C. N. WONG (Canada)

An experimental study of factors important in document ranking. 186
D. WILLIAMSON HARMAN (USA)

wednesday september 10, 1986

SESSION 9: KNOWLEDGE BASED INFORMATION RETRIEVAL (I)
Chairman: C. J. VAN RIJSBERGEN

INVITED PAPER - A new theoretical framework for Information Retrieval. 194
C. J. VAN RIJSBERGEN (Ireland)

User-specified domain knowledge for document retrieval. 201
B. CROFT (USA)

SESSION 10: KNOWLEDGE BASED INFORMATION RETRIEVAL (II)
Chairman: C. J. VAN RIJSBERGEN

IOTA: A full text Information Retrieval system. 207
Y. CHIARAMELLA, B. DEFUDE, M.F. BRUANDET, D. KERKOUBA (France)

An Information Retrieval system based on artificial intelligence techniques. 214
D. DE JACO, G. GARBOLINO (Italy)

The use of inference mechanisms to improve the retrieval facilities from large relational databases. 221
G. P. ZARRI (France)

SESSION 11: LEARNING SYSTEMS
Chairman: G. SALTON

A machine learning approach to Information Retrieval. 228
S. K. M. WONG, W. ZIARKO (Canada)

An automatic and tunable document indexing system. 234
E. OZKARAHAN, F. CAN (USA)

Performance of self-taught documents.
A. BOOKSTEIN (USA) 244

SESSION 12: PROBABILISTIC RETRIEVAL
Chairman: A. BOOKSTEIN

Two models of retrieval with probabilistic indexing. 249
N. FUHR (W. Germany)

Two Poisson and binary independence assumptions for probabilistic document retrieval. 258
R. LOSEE, A. BOOKSTEIN, C. T. YU (USA)

Non-binary independence model. 265
C. T. YU, T. C. LEE (USA)

The maximum entropy principle in Information Retrieval. 269
P. B. KANTOR, J. J. LEE (USA)

An interpretation of index term weighting schemes based on document components. 275
K.L. KWOK (USA)

INDICE AUTORI

N.J. BELKIN, (USA) 11
C. BERRUT (France) 23
A. BOOKSTEIN (USA) 244, 258
P. BOSC (France) 114
M.F. BRUANDET (France) 207
F. CAN (USA) 234
Y. CHIARAMELLA (France) 207
Y CHOUEKA (Israel) 8
M. COURANT (France) 114
B. CROFT (USA) 201
G.R. CROSS (USA) 144
D. CROUCH (USA) 58
C.G. DEBESSONET (USA) 144
B. DEFUDE (France) 207
D. DE JACO (Italy) 214
J. S. DEOGUN (Canada) 157
U. DEPPISCH (W. Germany) 77
M. K. DI BENIGNO (USA) 144
T. DOSZKOCS (USA) 49
A. EL-HAMDOUCHI (U. K.) 149
A. S. FRAENKEL (Israel) 88
N. FUHR (W. Germany) 249
S. GALLELLI (Italy) 31
G. GARBOLINO (Italy) 214
S. GIBBS (Switzerland) 23
C. IACOBELLI (Italy) 31
P. INGWERSEN (Denmark) 68
P. B. KANTOR (USA) 269
D. KERKOUBA (France) 207
S. T. KLEIN (Israel) 88
B. H. KWASNIK (USA) 11
K. L. KWOK (USA) 275
J. J. LEE (USA) 269
T. C. LEE (USA) 265
R. LOSEE (USA) 258
I. A. MACLEOD (Canada) 131
P. MARCHISIO (Italy) 31
P. MARTIN (Canada) 131
B. NORDIN (Canada) 131
E. OZKARAHAN (USA) 234
P. PALMER (France) 123
V. RAGHAVAN (Canada) 157, 175
T. RAITA (Finland) 97
M. RATHGEB (W. Germany) 39
S. ROBIN (France) 114
G. SALTON (USA) 1
E. SEGAL (Israel) 88
A. F. SMEATON (Ireland) 103
J. TEUHOLA (Finland) 97
G. THURMAIR (W. Germany) 138
W.G.T. TRUCKENMÜLLER (W. Ger.) 39
D. TSICHRITZIS (Switzerland) 23
C. J. VAN RIJSBERGEN (Ireland) 194
E. VOORHEES (USA) 164
P. WILLET (U.K.) 149
D. WILLIAMSON HARMAN (USA) 186
P. C. N. WONG (Canada) 175
S. K. M. WONG (Canada) 175, 228
I. WORMELL (Denmark) 68
C. T. YU (USA) 258, 265
G. P. ZARRI (France) 221
W. ZIARKO (Canada) 175, 228

Recent Trends in Automatic Information Retrieval

Gerard Salton*
Department of Computer Science
Cornell University
Ithaca, NY 14853

Abstract

Substantial successes were achieved in the early years in automatic indexing and retrieval using single term indexing theories with term weight assignments based on frequency considerations. The development of more refined indexing systems using thesaurus aids and automatically constructed term association maps changed the retrieval effectiveness only slightly. The recent introduction of the relevance concept in the form of probabilistic retrieval models provided a firm basis for term weighting and document ranking practices. However, the probabilistic methods were not helpful in substantially enhancing the retrieval effectiveness.

At the present time, attempts are made to add artificial intelligence concepts to the document retrieval environment in the form of fancy graphics interfaces, learning systems for query and document indexing and for collection searching, extended logic models relating documents and information requests, and analysis methods based on the use of semantic maps and other kinds of knowledge structures. Using the earlier developments and evaluation results as guidelines, an attempt is made to outline the information retrieval environment of the future and to assess the usefulness of some of the currently proposed search and retrieval methods.

1. Automatic Information Retrieval

Information retrieval deals with text analysis, text storage, and the retrieval of stored records that are similar in some sense to information requests received from a population of users. From the beginning, it was realized that some retrieval tasks would be more difficult to mechanize than certain others: whereas the computation of similarity coefficients between the content identifiers attached to stored records and to user information requests could be mechanized relatively easily, it was argued early on that the content analysis task itself would probably have to be performed by human beings and not by machines for a long time to come:

> "it is very likely that manual indexing (content analysis) by cheap clerical labor will still, on average, be qualitatively superior to any kind of automatic indexing....neither the assignment of topic terms to a given request, nor the reformulation of a request are processes which could conceivably be adequately mechanized, contrary to some speculation in this direction." [BARH62]

In spite of these predictions, efforts have continued over the years to devise workable automatic indexing methods, and effective as well as efficient automatic retrieval procedures. In the early years, the goodness of a text word for content identification purposes was thought to be dependent principally on the frequency of occurrence of a word in a particular text:

> "a notion occurring at least twice in the same paragraph would be considered a major notion. A notion which occurs in the immediately preceding or succeeding paragraph would be considered a major notion even though it appears only once in the paragraph under consideration. Notations for major notions as just defined would then be listed in some standard order as representative of that paragraph..." [LUHN57]

More recently, it was realized that words occurring frequently in the texts of particular documents could not be used to distinguish these documents from the remaining texts of a collection if their occurrence frequency was high also in all the other available documents. This led to the notion that a good term should have a high term frequency in a particular document, but a low overall frequency in the collection. [SPAR72] These insights were used to generate a term weighting formula, known as "tf × idf", consisting of the product of the term frequency inside a given document multiplied by the inverse document frequency representing a function inverse to the number of documents to which a term is assigned. [SALT73]

Somewhat surprisingly, the single term indexing theories in which document or query content is represented by weighted sets of single terms extracted from the corresponding document or query

*This study was supported in part by the National Science Foundation under grant IST-83-16166.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

texts proved to be quite powerful. Various experiments performed to compare manual content analysis with automatic single term indexing systems indicated in each case that quite simple automatic term extraction systems were not inferior in retrieval effectiveness to more conventional manual analysis methods. [SALT85a] A typical sample evaluation output is shown in Table 1 in terms of recall (the proportion of relevant materials retrieved) and precision (the proportion of retrieved materials that are relevant) for a collection of 44,000 document titles and abstracts in aeronautics used with 40 search requests. [CLEV77] The output of Table 1 shows that the manual (or intellectual) indexing system produced somewhat better search precision at the cost of substantially reduced search recall in comparison with the automatic term extraction system.

Content Analysis Method	Recall	Precision
Automatic natural language indexing (automatic text search of document titles and abstracts)	0.78	0.63
Controlled language manual indexing (indexing by trained personnel using a schedule of controlled terms	0.56	0.74

Comparison of Manual and Automatic Indexing
(recall and precision figures for
44,000 NASA documents in aeronautics
averaged for 40 search requests)

Table 1

Cleverdon, who was in charge of the NASA system test reaches the following conclusions based on the result of Table 1:

> "It appears impossible to reach any other conclusion than that, within the parameters of this test, natural language searching on titles and abstracts proved at least equal to, and probably superior to (manual) searching on controlled language terms." [CLEV77]

While the original single term automatic text analysis systems proved quite effective, it was obvious that the single term sets extracted from document texts could offer only a simplified picture of actual text meaning and content. The suggestion then arose that more refined content identifying units than single terms be used for content representation. In particular, it was believed that two types of vocabulary relationships should be taken into account in analyzing text content: [LYON68]

- the paradigmatic relations that cover term associations such as synonyms and hierarchical term inclusion relations, that always exist between particular terms, regardless of the context in which the terms are used (for example, the relation between "computer" and "calculator", or the relation between "Paris" and "France");

- the syntagmatic relations between terms in which the relationship between linguistic entities depends on the context in which the terms are used (for example, the relation between "united" and "states" in the phrase "United States".

Various methods are outlined in the literature designed to make use of paradigmatic and syntagmatic relations in text content analysis. The main possibilities consist in using a thesaurus of related terms, or alternatively in constructing so-called term association maps that are capable of identifying "associated" term sets, or finally in generating complex text identifying units such as term phrases consisting of combinations of single terms. The idea in each case is to take particular single terms and to use them as entry points for the identification of related notions from the thesaurus or from the term association map; alternatively the single terms are used as a starting points for the generation of phrases. The new, associated terms and phrases can then be added to the originally available content identifiers, or the new terms can replace originally available terms.

When the subject area is narrowly circumscribed, and knowledgeable subject experts are available, useful thesaurus arrangements can be manually constructed by human experts. Such thesauruses may then provide substantial enhancements in retrieval effectiveness. Table 2 shows the average search precision obtained at certain fixed recall points for a collection of 400 documents in engineering used with 17 search requests. In Table 2, the performance obtained by extracting thesaurus class entries from a manually constructed thesaurus (the Harris Three thesaurus) is compared with a simpler analysis system where the weighted terms obtained from query and document texts are used directly for content analysis purposes. The single term base case of Table 2 uses a term frequency (tf) term weighting system which normally performs better than an unweighted single term system, but not as well as the preferred tf × idf term weights.

The output of Table 2 shows that at the high recall end of the performance range, the thesaurus provides much better retrieval output than the weighted word stem process. The average advantage of the thesaurus when low recall points are also taken into account is 13 percent. [SALT68a] The use of thesauruses is widely advocated as a means for normalizing the vocabulary of document texts. However, the construction of useful thesauruses is an art rather than a science and requires extensive knowledge of the particular subject area under consideration and of the record collections to be processed. The construction of a thesaurus is thus a major undertaking which needs to be repeated for each distinct subject area. The so-called thesaurus method is therefore difficult to implement efficiently in operational environments.

Recall	Average Search Precision		
	Weighted Word Stems	Harris Three Thesaurus	Automatic Term Associations
.1	.9563	.9735 + 2%	.7385 -23%
.3	.7986	.8245 + 3%	.5844 -27%
.5	.6371	.7146 +11%	.5187 -19%
.7	.4877	.6012 +19%	.4452 - 9%
.9	.3426	.4973 +31%	.3794 +11%
Average Improvement		+13%	-13%

Sample Thesaurus and Associative Indexing Performance
(IRE collection, 400 documents, 19 queries)

Table 2

In the early years it was hoped that thesauruses could be built automatically by studying the occurrence characteristics of the terms in the documents of a collection, and grouping into common thesaurus classes those terms that would co-occur sufficiently often in the texts of the documents. Later it was recognized that thesauruses constructed by using the occurrence characteristics of the vocabulary in the documents of a collection do not in fact identify generally valid paradigmatic term relations, but provide instead locally valid syntagmatic relations derived from the vocabulary of the collections under consideration. [LESK69] Such locally valid term grouping systems are known as term association maps, and the indexing method which consists in adding associated terms obtained from a term association map to the originally available document terms is known as associative indexing. [DOYL61,GIUL62]

In practice, it is found that the use of term associations can improve the search recall by providing new matches between the terms assigned to queries and documents that were not available before the identification of the associated terms. In addition, the search precision can also be enhanced by reinforcing the strength of already existing term matches. [LESK69] Unfortunately, the experimental evidence indicates that only about 20 percent of automatically derived associations between pairs of terms are semantically valid. As a result, the associative indexing process provides many erroneous associations in addition to some useful ones and on average the associative methodology does not produce guaranteed advantages in retrieval effectiveness.

The right hand side of Table 2 contains evaluation results for the associative indexing method applied to the document collection in engineering previously used with the Harris Three thesaurus. The results of Table 2 show that the automatic term associations provide an increase in average search precision only at the high recall end of the performance range. Overall, the average search precision of the associative method decreases by 13 percent compared with the use of weighted single terms using term frequency weights. [SALT68b, p.130]

The use of manually constructed thesauruses and automatically derived term association maps is designed principally to enhance search recall by adding new related terms to the terms originally available to identify document and query content. Search precision may be enhanced by using narrow, specific content identifiers instead of broader, less specific terms. This suggests that broad single terms, exhibiting high occurrence frequencies in the documents of a collection be replaced by narrower term phrases. For example, a broad term such as "computer" might be replaced by narrower constructions such as "computer system", or "computer programming".

In principle, the identification of useful term phrases must be based on syntactic language analysis systems, supplemented by semantic know-how for the subject area under consideration. Unfortunately, complete linguistic analysis methods covering topic areas of reasonable scope are not available for operational use. In practice, it is therefore necessary to fall back to simpler methodologies in which phrases are identified as sequences of terms that co-occur in certain contexts in the documents and queries of a collection. Because phrase formation is a term narrowing operation, one normally insists that one component of the phrase (the phrase head) be a high-frequency component in the collection under consideration. Furthermore, a domain of co-occurrence must be specified between the phrase components that are to be included in a phrase: typically, a phrase may be formed when two particular terms occur in the same sentence of a document or query, and when the number of intervening words does not exceed a stated small number.

The evaluation results for a typical nonsyntactic phrase formation process are shown in Table 3 for two collections of documents in computer science (the CACM collection) and documentation (the CISI collection) averaged over 52 and 76 user queries respectively. The base run for the output of Table 3 is a single term assignment process where the terms are assigned tf × idf weights. The phrase process used for the experiments of Table 3 consisted in adding phrases to the available single terms whenever pairs of adjacent word stems would occur in the same sentences of documents and queries. The phrase weight used is, in each case, the average of the normalized tf × idf weights of the individual phrase components. [FAGA85]

The output of Table 3 shows that the phrase formation process operates as expected for the CACM collection since the narrowing of the indexing vocabulary enhances the search precision at the low recall end of the performance range. Overall the advantage of the phrase formation process is about 8 percent in average precision at the 5 recall points for the CACM collection. The same phrase process unfortunately does not work well for the CISI collection in documentation; in that case the performance with and without phrases is approximately the same. For CISI, the presence of good, semantically acceptable phrases that might serve in improving retrieval performance is evidently compensated by the formation of many adjacent matching word groups that are extraneous and hence are unable to pull out relevant materials when they happen to be present both in the queries and documents of a collection.

Recall	CACM (3204 docs, 52 queries)			CISI (1460 docs, 76 queries		
	Single Terms	Single Terms plus Phrases		Single Terms	Single Terms plus Phrases	
.1	.5086	.5580	+10%	.4919	.4813	- 2%
.3	.3672	.4065	+11%	.3118	.3158	+ 1%
.5	.2398	.2835	+18%	.2320	.2291	- 1%
.7	.1462	.1466	0%	.1504	.1463	- 3%
.9	.0711	.0704	- 1%	.0739	.0717	- 3%
Average Improvement			+7.6%			-1.6%

Average Search Precision for Single Term Indexing
with Nonsyntactic Phrase Indexing for
Four Sample Document Collections

Table 3

The evaluation results presented in Tables 1 to 3 indicate that the available automatic term extraction and term weighting methods are surprisingly effective in retrieving a large proportion of the relevant materials included in various document collections, and in rejecting a large proportion of the nonrelevant items. In general, the single term automatic indexing methods are easily competitive with manual content analysis methods based on the intellectual efforts of trained indexers. On the other hand, attempts to improve the recall performance of the single term methods by adding related terms derived from thesauruses and word association maps, or to enhance the precision by adding automatically determined term phrases, are much less successful.

The procedures designed to generate the complex indexing units in the form of thesaurus classes or term phrases are obviously not sufficiently discriminating to insure that a large proportion of these more refined content identification units are in fact appropriate for the collections under consideration. The manual thesaurus construction process is not easily replicated for different subject areas, and the phrase formation process may require additional syntactic and semantic controls to insure that the number of extraneous identifiers that are detected remains limited. For present purposes it appears reasonable to stay with the simpler single-term indexing methods.

2. The Introduction of Term Relevance

The notion of term relevance is absent from the methods examined in the previous section. That is, no distinction is made between terms that occur in documents that are either relevant, or nonrelevant, to particular queries. Instead the term values are determined by the term occurrence characteristics in the whole document collection regardless of the relevance properties of the individual items.

A number of retrieval models have been introduced in which the concept of document or term relevance takes a central place. The best known of these is the probabilistic retrieval model. [BOOK75, COOP78,HART75,KRAF78,ROBE83,VANR79] In the probabilistic model, the retrieval operation consists in ranking the documents of a collection in decreasing order of the ratio of their probability of being relevant to a query to the probability of their being nonrelevant. Given a document $\underline{x}$, it then becomes necessary to estimate the probabilities $P(\underline{x}|rel)$ and $P(\underline{x}|nonrel)$. These probabilities can in turn be made to depend on the probabilities of relevance of the individual terms x_i occurring in the respective documents. When one assumes that the individual terms occur independently of each other in the relevant and the nonrelevant documents of a collection, an optimum term weight w_i can be derived for each term x_i occurring in an information request which is equal to

$$w_i = \log \frac{P(x_i|rel)[1-P(x_i|nonrel)]}{P(x_i|nonrel)[1-P(x_i|rel)]} \qquad (1)$$

Assuming that the individual term relevance weights, w_i are available, each document can then be assigned a global document relevance score equal to the sum of the w_i factors for all query terms that are also present in each particular document. This provides the needed document ranking function.

The probabilistic retrieval model is attractive because it provides a theoretical foundation for the retrieval operation which takes into account the notion of document relevance. Furthermore, it is possible in the probabilistic environment to take into account at least some of the dependencies and relationships between the terms used to identify the queries and the stored records. [HARP78,VANR77] Finally, the probabilistic model offers justification for various methods that had previously been used in automatic retrieval environments on an empirical basis. For example, the useful inverse document frequency (idf) term weighting system, which ranks the terms in decreasing order of the total number of documents in a collection to which the terms are assigned, can be shown under appropriate assumptions to be similar to the term relevance weight of expression (1). [WUSA81]

Unfortunately, the probabilistic methodology has been disappointing in practice for a variety of reasons.

- the basic probabilistic methodology rests on the use of initially unweighted document and query terms, even though it is known that substantial differences exist in the worth of individual terms for content identification purposes;

- the computation of the term relevance weights depends on knowledge about the relevance of individual documents to information requests; since this relevance information is normally not available at search time, the probabilistic weights can be estimated only imperfectly;

- too many subsets of terms with potential dependencies between them exist to make it practical to include all of them in any retrieval model; it is then necessary to use simplified term dependency models from which some useful information will necessarily be missing.

The result is that the probabilistic model has been important mainly for providing solid foundations for much of the information retrieval work rather than as a practical tool in experimental or operational situations.

This last point may be illustrated by considering the results of term weighting experiments carried out recently by Christopher Buckley at Cornell University, where 154 different term weighting methods were processed against six different document collections. [BUCK85] The different weighting system were obtained by considering various forms of the term frequency factor (tf) of a term, the inverse document frequency factor (idf) of the term, and the normalization method, if any, used to compute the total term weights. In practice, different term weighting systems are usable for query terms and document terms, and the normalization factor applied to the weights may be used to simulate various query-document matching methods.

Consider a particular document $D = (w_{d_1}, w_{d_2}, \ldots, w_{d_t})$ and query $Q = (w_{q_1}, w_{q_2}, \ldots, w_{q_t})$, where w_{d_k} and w_{q_k} represent the weights of the kth terms in D and Q respectively. The similarity between D and Q may then be computed as

$$sim(D,Q) = \sum_{k=1}^{t} w_{d_k} \cdot w_{q_k} . \qquad (2)$$

For unnormalized term weights, expression (2) represents simply the vector product between the vector representations of D and Q. On the other hand, when a cosine normalization is used for w_{d_k} and w_{q_k}, then expression (2) becomes a cosine computation. For example, when standard term frequency weights are in use, $\sum_{k=1}^{t} tf_{d_k} \cdot tf_{q_k}$ represents a normal weighted vector product. However, with weighted term frequency factors $tf_{d_k} / \sqrt{\sum_{i=1}^{t} tf_{d_{k_i}}}$ and $tf_{q_k} / \sqrt{\sum_{i=1}^{t} tf_{q_i}}$, the formula of expression (2) produces a cosine measure.

Table 4 shows the average precision results for 6 particular term weighting systems, including two probabilistic weighting systems (termed best probabilistic term weighting and probabilistic binary term independence, respectively), three standard term weighting methods (termed best tf × idf, normal idf, and normal tf, respectively), and finally a coordination level match corresponding to the absence of any weighting before using expression (2) to compare queries and documents. For the probabilistic term weights, approximations are used which make it possible to compute the formula of expression (1) without knowing any relevance information [CROF79]. The run corresponding to "best probabilistic weight" uses the normal probabilistic approximation for query terms (that is, $\log(N-n_i)/n_i$, where N is collection size and n_i is the number of documents with term i), and a term frequency weight of limited range defined as (0.5 + 0.5 tfi/max tf) for document terms. The query-document similarity measure for the "best probabilistic weight" system is then computed as

$$sim(D,Q) = \sum_{k=1}^{t} \left[0.5 + \frac{0.5\ tf_{d_k}}{\max tf_d} \right] \log \left(\frac{N-n_{q_k}}{n_{q_k}} \right) \qquad (3)$$

Analogously the weighting factors being multiplied for the "best (tf × idf)" run are respectively

$$\frac{\left(tf_{d_k} \cdot \log \frac{N}{n_{d_k}} \right)}{\sqrt{\sum_{k=1}^{t} \left(tf_{d_k} \log \frac{N}{n_{d_k}} \right)^2}} \quad \text{and} \quad \left(0.5 + \frac{0.5\ tf_{q_k}}{\max tf_q} \right) \log \frac{N}{n_{q_k}} . \qquad (4)$$

In addition to the average precision values for the six weighting systems, Table 4 also shows the ranks out of 154 for the given weighting methods used with each collection. In each case, a rank of 1 designates the best weighting system for each collection, and 154 the worst. The result of Table 4 indicates that the (tf × idf) weighting systems are generally better than the best probabilistic methods; these in turn are better than the standard idf weighting systems, the conventional binary term independence methods, the normal tf methods, and finally the coordination level matches, in that order. The average rank of the best (tf × idf) method is 17.8 for the six collections whereas the average rank of the coordination level match is 117.1. An alternative weighted probabilistic system proposed by Croft does not produce result that are as good as those of the best probabilistic system of expression (3). [CROF81,CROF83]

The results of Table 4 show that the basic ranking of the various weighting systems is the same for all collections with the exception of the NPL collection where the probabilistic weighting systems are better than the other frequency based weights. Experimental results based on the NPL collection were used earlier to claim superiority for the probabilistic methodology. [CROF84] However, as Table 4 shows, the NPL records represent a special case. The NPL document and query representations are very short and a large mass of low frequency terms are used to index the collection. In fact, most NPL terms exhibit a frequency of 1 in the collection; the normal term frequency weights are

Collection	Best tf × idf		Best probabilistic term weighting		Normal idf	
	Rank	Average Precision	Rank	Average Precision	Rank	Average Precision
CACM 3204	7	.3130	34	.2713	55	.2385
CISI 1460	11	.1874	73	.1377	86	.1314
MED 1033	5	.5628	26	.5443	84	.5038
CRAN 1398	22	.3821	27	.3772	98	.3174
INSPEC 12684	7	.2626	42	.2092	72	.1779
NPL 11429	55	.1933	2	.2755	25	.2401
Average rank	17.8		34.0		70.0	

Collection	Probablistic binary term independence		Normal tf		Coordination Level	
	Rank	Average	Rank Precision	Average	Rank Precision	Average
CACM 3204	78	.2197	93	.1910	104	.1721
CISI 1460	99	.1237	109	.1196	134	.1019
MED 1033	75	.5072	119	.4641	148	.4070
CRAN 1398	100	.3154	74	.3371	147	.2370
INSPEC 12684	92	.1549	83	.1621	130	.0970
NPL 11429	12	.2583	83	.1749	40	.2094
Average rank	76.0		93.5		117.1	

Average Precision for Various Term Weighting Systems
for Six Document Collections

Table 4

therefore not effective in that case. The best probabilistic term weighting system is ranked 34th out of the 154 methods considered.

Another possibility for introducing the effect of document and term relevance into the retrieval process is the well-known relevance feedback process. [ROCC71,IDEE71] In that case, the results of an initial search run are used automatically to reformulate the search request by increasing the weights of query terms that are present in previously retrieved documents termed relevant to the query, and contrariwise in decreasing the weights of query terms also present in the nonrelevant documents previously retrieved. In addition to changing the query term weights, the queries can also be expanded by adding new terms to the query formulations taken from the relevant documents previously retrieved. The relevance feedback process is normally applied to vector queries where each query is formulated on a set of query terms. However methods are also available for using relevance feedback with Boolean queries. [SALT85b]

Table 5 shows typical relevance feedback results for one iteration of feedback -- that is, initial search followed by a single query reformulation -- for the two collections previously used for the experiments of Table 3. As usual, average precision results are shown for the 52 CACM queries and the 76 CISI queries for feedback systems with and without query term expansion. The base run used for comparison purposes is a (tf × idf) term weighting system with a cosine similarity normalization used for query and document terms. Fifteen initially retrieved documents are examined for relevance in the runs of Table 5, and all relevant retrieved items plus the top nonrelevant retrieved item are used for feedback purposes (the so-called "Dec Hi" method [IDEE71]).

The output of Table 5 shows that one iteration of relevance feedback produces over 90 percent improvement in average precision for CACM and nearly 50 percent for CISI, thus confirming the usefulness of query reformulation methods that are based on simple user input such as relevance judgments for previously retrieved items. When relevance information is available, useful performance improvements are thus obtainable with quite simple iterative search systems.

Recall	CACM 3204, 52 queries			CISI 1460, 76 queries		
	Initial tf × idf cosine	First Feedback Iteration without expansion	with expansion	Initial tf × idf cosine	First Feedback Iteration without expansion	with expansion
.1	.3641	.3546 (- 3)	.5338 (+47)	.2476	.3631 (+47)	.3654 (+48)
.3	.2038	.2633 (+29)	.3579 (+76)	.1500	.2369 (+58)	.2899 (+93)
.5	.1329	.1743 (+31)	.2804 (+111)	.1209	.1729 (+43)	.1647 (+36)
.7	.0732	.1038 (+42)	.1621 (+121)	.0753	.1154 (+53)	.1097 (+46)
.9	.0405	.0542 (+35)	.0827 (+104)	.0505	.0644 (+28)	.0611 (+21)
Average Improvement		+26.6%	+91.8%		+45.8%	+48.8%

One Iteration of Relevance Feedback with and without Query Expansion (initial run: tf × idf, cosine similarity; feedback method: Ide Dec-Hi method)

Table 5

3. Future Retrieval Developments

The evaluation studies described in the preceding sections indicate that the currently preferred automatic retrieval methodology consists in using selected, properly weighted single terms for the representation of text content, and conducting iterative searches in which query statements are successively refined under user control. [IDEE71,PORT82,ROCC71,SALT85b,VERN77] The recent emphasis on hands-on computing, fancy workstations, and computer networking suggests that improved retrieved strategies could be implemented by making use of the available know-how of system users and retrieval experts. The following possibilities must then be considered in the development of future retrieval technologies:

- the use of advanced user-system interfaces and graphic displays;

  the use of learning approaches in which the system operations can be improved based on experience acquired during earlier search operations;

- the use of soft, or fuzzy matching systems designed to assess the similarities between incompletely matching information queries and stored records;

- the introduction of expert system methods based on human know-how and previously stored knowledge.

Concerning first the use of advanced display systems, it is obvious that documents identified by sets of weighted terms can be represented as vectors in a multidimensional vector space, where similar documents will be situationed close together. Relationships between documents and terms assigned to them can similarly be represented by appropriate graph structures, and term thesaurus relationships are specifiable by term association maps. A particular information search is then carried out by navigation in the document space. A good deal of effort has been devoted to the design of display systems in the hope of simplifying the information space manipulations. At the present time, very little is however known about which type of displays and interactions are most effective. [CROU86,DOCZ86,INGW86]

Learning techniques are already incorporated in existing retrieval systems in the form of iterative search systems and relevance feedback techniques. Related methods may lead to the generation of improved document (as opposed to query) representations using various kinds of document space modification methods. [BOOK86,BRAU71,KERC71] Learning techniques are also usable in the design of document and term classes or clusters. Adaptive document clusters have been generated based on available document relevance assessments [RAGH86,YUWA85], and so-called pseudo-classification methods have been used for automatic term grouping. [SALT80] Learning methods have also been proposed for indexing and retrieval [WONG86], and to determine criteria for document retirement. [SARD74] In all these cases, the accuracy of the methodology depends on the representations of the document set constituting the learning sample, and on the degree to which the earlier user environment and the earlier search strategies are also appropriate at a later time.

It is well known that in information retrieval an exact coincidence between stored records and available search requests is rarely present. In these circumstances it is necessary to perform partial, or incomplete matches to identify items that are approximately similar to the query statements. In vector representations where queries and documents are represented by sets of content terms, similarity measures such as the cosine coefficient can be used to obtain global query-document similarity indications. To compute the similarity between Boolean queries and stored document vectors, an extended Boolean logic was recently introduced which accommodates term weights attached to the Boolean query terms, and well as strictness parameters attached to the Boolean operators. A typical query might then be processed as

$$\{[(A,a) \underline{\text{or}}^{p_1} (B,b)] \underline{\text{and}}^{p_2} (C,c)\}$$

where the values of p_1 and p_2 determine the strength of interpretation of the corresponding operators or and and. and a, b, and c are the weights attached to terms A, B, and C. In the extended Boolean system, the similarity coefficients between a document $D = (A,d_A; B,d_B)$ and queries $Q_1 = [(A,a)\ \underline{or}^p\ (B,b)]$ and $Q_2 = [(A,a)\ \underline{and}^p\ (B,b)]$ are computed respectively as

$$sim(D,Q_1) = \left[\frac{a^p d_A^p + b^p d_B^p}{a^p + b^p} \right]^{1/p}$$

$$\text{and } sim(D,Q_2) = 1 - \left[\frac{(1-d_A^p)a^p + (1-d_B^p)b^p}{a^p + b^p} \right]^{1/p} \quad (5)$$

It is known that the extended Boolean system produces superior retrieval effectiveness. [SALT83]

Many other systems of multi-valued logic can provide multiple or even continuous matching values between queries and documents. In one proposal based on the use of minimal modification operators needed to achieve coincidence between a document and query, a system of conditional logic is used. [VANR86] All these new systems of nonclassical logic remain to be evaluated in a retrieval setting.

A final development in the design of novel retrieval environments is the proposed use of expert-system techniques. Expert systems are designed to provide the high-level know-how of human experts to control the operations of particular systems. To provide such expert assistance in an automatic environment two main types of information must be supplied: on the one hand, it is necessary to formulate the rules used by (human) experts to reach particular decisions, and on the other hand the knowledge used by the experts to reach decisions must be formalized and stored. This leads to the design and storage of knowledge bases including both the specialized knowledge that relates to a particular subject areas and particular sets of tasks to be accomplished, as well as the every-day common sense knowledge that is assumed to be accessible to everyone.

Knowledge bases are normally built up from entities of various kinds supplemented by relationship indicators between particular types of entities. Many proposals exist for the design and implementation of knowledge bases; however, a consensus about the best way of doing the job is lacking at the moment. One possibility consists in asking the user population for advice about the needed knowledge, and in observing the rules by which human retrieval experts actually carry out their tasks. [CROF86,FIDE86] Whether such an approach is fruitful remains to be seen.

The existing efforts with expert system approaches consist mostly in the design of high-level dialogues between system and users to elicit from the user population the information needed to reach particular decisions regarding desirable search and retrieval activities. [GUID83,ODDY77,POLL83] It is unclear what type of user-system interaction is most effective in this respect, nor is it known whether system users can in fact contribute meaningfully to the construction of knowledge bases or search and retrieval rules. When the subject areas under consideration and the domain of coverage are sufficiently restricted, useful knowledge structures and appropriate processing rules can probably be generated by human specialists in each particular topic area. [DIBE86] Whether generally replicable expert system procedures can be designed to serve outside very restricted topic slices remains to be seen. Until the design of the newer technologies is better understood, and the corresponding procedures will be evaluated, it seems reasonable to remain cautious about new retrieval environments.

References

[BARH62] Bar Hillel Y., "Theoretical Aspects of the Mechanization of Literature Searching", in Digitale Informationswandler, W. Hoffman, editor, Vieweg and Sons, Braunschweig, 1962, 406-443.

[BOOK75] Bookstein A., D. Swanson D., "A Decision Theoretic Foundation for Indexing", Journal of the ASIS, 26:1, 1975, 45-50.

[BOOK86] Bookstein A., "Performance of Self-Taught Documents", 1986 ACM Conference on Research and Development in Information Retrieval, Pisa, Italy, September 1986.

[BRAU81] Brauen T.L., "Document Vector Modification", in The Smart Retrieval System: Experiments in Automatic Document Processing, G. Salton, editor, Prentice Hall Inc., Englewood Cliffs, NJ, 1971, 450-484.

[BUCK85] Buckley C., "Notes on Automatic Term Weighting and Relevance Feedback", Department of Computer Science, Cornell University, Ithaca, NY, May 1985.

[CLEV77] Cleverdon C.W., "A Computer Evaluation of Searching by Controlled Language and Natural Language in an Experimental NASA Data Base", European Space Agency, Report ESA 1/432, Frascati, Italy, July 1977.

[COOP78] Cooper W.S., Maron M.E., "Foundations of Probabilistic and Utility-Theoretic Indexing", Journal of the ACM, 25:1, 1978, 67-80.

[CROF79] Croft W.B., Harper D.J., "Using Probabilistic Models in Information Retrieval Without Relevance Information", Journal of Documentation, 35, 1979, 285-295.

[CROF81] Croft W.B., "Document Representation in Probabilistic Models of Information Retrieval", Journal of the ASIS, 32:6, 1981, 451-457.

[CROF83] Croft W.B., "Experiments with Representation in a Document Retrieval System", Information Technology, 2:1, January 1983, 1-22.

[CROF84] Croft W.B., "A Comparison of the Cosine Correlation and the Modified Probabilistic Model", Information Technology, 3:2, April 1984, 113-114.

[CROF86] Croft W.B., "User Specified Domain Knowledge for Document Retrieval", 1986 ACM Conference on Research and Development in Information Retrieval, Pisa, Italy, September 1986.

[CROU86] Crouch D.C, "The Visual Display of Information in an Information Retrieval Environment", 1986 ACM Conference on Research and Development in Information Retrieval, Pisa, Italy, September 1986.

[DIBE86] diBenigno M.K., Cross G.S., de Bessonet C.G., "COREL - A Conceptual Retrieval System", 1986 ACM Conference on Research and Development in Information Retrieval, Pisa, Italy, September 1986.

[DOCZ86] Doczkocs T.E., "Information Retrieval - Natural Language Processing - Artificial Intelligence - User Friendly On-Line Systems", 1986 ACM Conference on Research and Development in Information Retrieval, Pisa, Italy, September 1986.

[DOYL61] Doyle L.B., "Semantic Road Map for Literature Searches", Journal of the ACM, 8, 1961, 553-578.

[FAGA85] Fagan J.L., "Automatic Phrase Indexing for Text Passage Retrieval and Printed Subject Indexes", Technical Report, Department of Computer Science, Cornell University, Ithaca, NY, May 1985.

[FIDE86] Fidel R., "Toward Expert Systems for the Selection of Search Keys", Journal of the ACM, 37:1, 1986, 37-44.

[GIUL62] Giuliano V.E., "Automatic Message Retrieval by Associative Techniques", in Joint Man Computer Languages, Mitre Corporation, Report SS-10, Bedford, MA, 1962, 1-44.

[GUID83] Guida G., Tasso C., "IR-NLI: An Expert Natural Language Interface to On Line Databases", Conference on Applied Natural Language Processing, Association for Computational Linguistics, Santa Monica, CA, 1983, 31-38.

[HARP78] Harper S.P., van Rijsbergen C.J., "An Evaluation of Feedback in Document Retrieval Using Cooccurrence Data", Journal of Documentation, 34:3, September 1978, 189-206.

[HART75] Harter S.P., "A Probabilistic Approach to Automatic Keyword Indexing", Parts 1 and 2, Journal of the ASIS, 26:4, 1975, 197-206 and 26:5, 1975, 280-289.

[IDEE71] Ide E., "New Experiments in Relevance Feedback", in The Smart System - Experiments in Automatic Document Processing, G. Salton, editor, Prentice Hall Inc., Englewood Cliffs, NJ, 1971, Chapter 16.

[INGW86] Ingwersen P., Worwell I., "Improved Subject Access Browsing and Scanning Mechanisms in Modern On-Line Information Retrieval", 1986 ACM Conference on Research and Development in Information Retrieval, Pisa, Italy, September 1986.

[KERC71] Kerchner M.D., "Dynamic Document Processing in Clustered Collections", Ph.D. Dissertation, Report ISR-19, Department of Computer Science, Cornell University, Ithaca, NY, October 1971.

[KRAF78] Kraft D.H., Bookstein A., "Evaluation of Information Retrieval Systems: A Decision Theory Approach", Journal of the ASIS, 29:1, 1978, 31-40.

[LESK69] Lesk M.E., "Word-Word Associations in Document Retrieval Systems", American Documentation, 20:1, January 1969, 27-38.

[LUHN57] Luhn H.P., "A Statistical Approach to Mechanized Encoding and Searching of Literary Information", IBM Journal of Research and Development, 1:4, October 1957, 309-317.

[LYON68] Lyons J., "Introduction to Theoretical Linguistics", Cambridge University Press, Cambridge, England, 1968.

[ODDY77] Oddy R.N., "Information Retrieval through Man-machine Dialogue", Journal of Documentation, 33, 1977, 1-14.

[POLL83] Pollitt S.E., "End User Searching for Cancer Therapy Literature - A Rule Based Approach", Proc. of Sixth Int. ACM SIGIR Conference, SIGIR Forum, 17:4, Summer 1983, 136-145.

[PORT82] Porter M.F., "Implementing a Probabilistic Information Retrieval System", Information Technology: Research and Development, 1:2, 1982, 131-156.

[RAGH86] Raghavan V.V., Deogun J.S., "User Oriented Document Clustering: A Framework for Learning in Information Retrieval", 1986 ACM Conference on Research and Development in Information Retrieval, Pisa, Italy, September 1986.

[ROBE83] Robertson S.E., Maron M.E., Cooper, W.S., "The Unified Probabilistic Model in Information Retrieval, in Research and Development in Information Retrieval", G. Salton and H.J. Schneider, editors, Lecture Notes in Computer Science, Vol. 146, Springer-Verlag, 1983, 108-117.

[ROCC71] Rocchio Jr. J.J., "Relevance Feedback in Information Retrieval", in The Smart System - Experiments in Automatic Document Processing, Prentice Hall Inc., Englewood Cliffs, NJ, 1971, Chapter 14.

[SALT68a] Salton G., Lesk M.E., "Computer Evaluation of Indexing and Text Processing", Journal of the ACM, 15:1, January 1968, 8-36.

[SALT68b] Salton G., "Automatic Information Organization and Retrieval", McGraw Hill Book Co., New York, 1968.

[SALT73] Salton G., Yang C.S., "On the Specification of Term Values in Automatic Indexing", Journal of Documentation, 29:4, December 1973, 351-372.

[SALT80] Salton G., "Automatic Term Class Construction Using Relevance - A Summary of Work in Automatic Pseudoclassification", Information Processing and Management, 16, 1980, 1-15.

[SALT83] Salton G., Fox E.A., Wu H., "Extended Boolean Information Retrieval", Communications of the ACM, 26:11, November 1983, 1022-1036.

[SALT85a] Salton G., "Another Look at Automatic Text Retrieval Systems", Technical Report TR 85-713, Department of Computer Science, Cornell University, Ithaca, NY, December 1985; to be published in Communications of the ACM, 29:7, July 1986.

[SALT85b] Salton G., Fox E.A., Voorhees E., "Advanced Feedback Methods in Automatic Information Retrieval", Journal of the ASIS, 36:3, 1985, 200-210.

[SARD74] Sardana K., "Automatic Document Refinement Algorithms", Report No. ISR-22, Section XII, Department of Computer Science, Cornell University, Ithaca, NY, November 1974.

[SPAR72] Sparck Jones K., "A Statistical Interpretation of Term Specificity and its Application in Retrieval", Journal of Documentation, 28:1, March 1972, 11-21.

[VANR77] van Rijsbergen C.J., "A Theoretical Basis for the Use of Cooccurrence Data in Retrieval", Journal of Documentation, 33, 1977, 106-119.

[VANR79] van Rijsbergen C.J., "Information Retrieval", Second Edition, Butterworths, London, 1979.

[VANR86] van Rijsbergen C.J., "A Nonclassical Logic for Information Retrieval", 1986 ACM Conference on Research and Development in Information Retrieval, Pisa, Italy, September 1986.

[VERN77] Vernimb C., "Automatic Query Adjustment in Document Retrieval" Information Processing and Management, 13, 1977, 339-353.

[WONC86] Wong S.K.M., Ziarko W., "A Machine Learning Approach in Information Retrieval", 1986 ACM Conference on Research and Development in Information Retrieval, Pisa, Italy, September 1986.

[WUSA81] Wu H., Salton G., "A Comparison of Search Term Weighting: Term Relevance vs. Inverse Doc. Frequency", ACM SIGIR Forum, 16:1, Summer 1981, 30-39.

[YUWA85] Yu C.T., Wang Y.T., Chen C.H., "Adaptive Document Clustering", Proc. of Eighth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, Association for Computing Machinery, Montreal, Canada, June 1985, 197-203.

USING STRUCTURAL REPRESENTATIONS OF ANOMALOUS STATES OF KNOWLEDGE FOR CHOOSING DOCUMENT RETRIEVAL STRATEGIES*

N.J. Belkin & B.H. Kwaśnik
School of Communication, Information and Library Studies
Rutgers University
New Brunswick, N.J., U.S.A.

ABSTRACT

We report on a project which attempts to classify representations of the anomalous states of knowledge (ASKs) of users of document retrieval systems on the basis of structural characteristics of the representations, and which specifies different retrieval strategies and ranking mechanisms for each ASK class. The classification and retrieval strategy specification is based on 53 real problem statements, 35 of which have a total of 250 evaluated documents. Four facets of the ASK structures have been tentatively identified, whose combinations determine the method and order of application of five basic ranking strategies. This work is still in progress, so results presented here are incomplete.

1. Introduction

It has been suggested for some time in the IR literature that different types of user situations, problems, goals, characteristics or questions might require different types of retrieval strategies, mechanisms, or ranking rules [e.g. BELK80; CROF84; ODDY77]. All such suggestions must address two major questions: how can different user situations be distinguished from one another? and, what kinds of retrieval strategies are appropriate to the different situations? To date, these remain open questions.

One previous study [BELK82] had suggested that structured representations of IR system users' anomalous states of knowledge (ASKs) might be used as the basis for choosing different document retrieval strategies. In [BELK82] and [HAPE85], some potential categorizations of ASKs and retrieval strategies were discussed; here we report on the preliminary results of an empirical classification, based on representations of 53 ASKs and about 250 documents which were evaluated by users in respect of those ASKs.

2. Methods

2.1 Data collection

Our data consists of narrative problem statements gathered from users of operational online document retrieval services, and of evaluations by those users of the usefulness of up to 15 documents in the resolution or management of their problem. Our methods for eliciting problem statements and evaluations are described in detail in HAPE85. Briefly, we collected our data from users of two academic information retrieval services of the University of London as they entered the service, but before they had spoken with the intermediary. The subjects were given a printed problem statement elicitation (figure 1) also posed to them orally. The oral elicitation and the user's narrative problem statement response were tape recorded. For one-half of the subjects, this tape recorded problem statement was then given to the intermediary, and used as the sole basis of the online search (non-interactive mode). In this case, the intermediary conducted the search alone. For the others, the problem statement was used as the basis for subsequent presearch interaction between the user and the intermediary (interactive mode). In this case, the user was with the intermediary throughout the search. In interactive mode searches, a check was made at the end of the interaction as to whether the original problem statement was still perceived valid by the subject.

1. Please give a clear indication of the research that you are doing at the moment. What is the nature of the research, its present stage of development and the research goals which you consider to be the most relevant to your information enquiry?
2. What is the information problem that has prompted you to have an online search carried out? Your answer should be a concise description of what it is you need to find out, rather than just a list of keywords.
3. What kinds of information would you like to receive as a result of the online search? For example: document type, the time period involved, the level of treatment, the breadth of coverage, language or languages, etc.

Figure 1. Problem statement elicitation

*Research supported by British Library Research and Development Department Grant SI/G/566 to the Department of Information Science, The City University, London

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.
© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

The results of the searches were sent to the subjects together with an evaluation questionnaire and transcript of their problem statement. The subjects were asked to evaluate up to 15 documents which they had read, according to their degree of usefulness with respect to the problem statement, and to comment on why they made each particular usefulness judgment. The transcripts of the problem statements, and the texts of the evaluated abstract documents were the basic data for input to the structural analysis program.

2.2 Text analysis

The problem statements were transcribed from the audio tapes according to a set of transcription rules developed for a series of discourse analysis projects at The City University [BROO83] [DANI85]. The transcript retains indications of pauses, false starts and other discourse phenomena, and represents words more-or-less as they were spoken. For the ASK representation programs, the raw transcripts were normalized to standard English narrative, primarily by removing indications of non-linguistic discourse phenomena and obvious re-start repetitions, and by using standard spellings. Sentence boundaries were also inserted according to rules based on length of pauses and discourse intonation.

The text analysis programs are described in detail in HAPE85. Their aim is to achieve graphical representations of both problem statements and abstracts, in which the nodes are concepts (represented by word stems), and the arcs indicate levels of association strength between nodes, with the distance between nodes also being an indication of their strength of association. The algorithm first applies a stop-list to the text, then a stemming procedure [PORT80], and then computes cumulative association strength for word pairs on the following conditions:

WORD-PAIR POSITION	SCORE
ADJACENT	12
SAME SENTENCE	4
ADJACENT SENTENCES	3

ASSOCIATION STRENGTH = SCORE

This association strength is treated as an inverse distance measure in a program written by John Bovey, which computes a stable two-dimensional network for the top 40 (or so) associates, according to the requirements for the graphs specified above. At the representation level, the association strengths are converted to four levels of strength, determined by the percentage contribution they make to the total association strength. Figure 2 is an example problem statement text, and figure 3 the corresponding graphical ASK representation. This general algorithm and representation was tested for adequacy by BELK82, and modified to its present configuration according to results from WEST83. Further work on its psychological validity is underway at Syracuse University [PALM84].

> The second topic is related to bleeding in early pregnancy and its effect on the outcome of that pregnancy. There are many studies actually carried out in Britain and in other countries on the effects of bleeding in early pregnancy on both the mothers and the foetuses. And little valid information has been obtained for these many studies, simply because ultrasound has not been used as a method of investigating the site of placents, so what we did, actually we did a sort of case control study of mothers with bleeding in early pregnancy compared with normal mothers, that's to say with no bleeding in early pregnancy. And we followed them during the whole period of pregnancy and we did subsequent type of ultrasound to both cases and controls and we compared between the outcomes of the two groups.
>
> I just want or would like to see - I mean this is answering question number 1 and answering question 2 - I would like to see other studies or similar studies elsewhere. As far as I know there are two studies, which I was able to take from Index Medicus, and I would like to see some more studies, if there is any possibility and comparing their approach. It's similar to the problem number 1. Yes, I want a document type on the printout for the ones which I can not get any access to - journals or books.

Figure 2. Text of the problem statement of s.14.

2.3 Characterizing ASK structures

BELK82 suggested that purely structural features of ASK representations could be used to classify the ASKs into groups which would each determine some specific, different retrieval strategy, or matching formula. These features were unspecified, however. We have developed a scheme for characterizing the ASK representations on the dimensions indicated in figure 4, which seem reasonable candidates for appropriate features.

GROUPS PRESENT IN STRUCTURE
- CLUSTERS (BY TYPE, MAGNITUDE & CONNECTIVITY)
- STARS (BY TYPE, MAGNITUDE & DEGREE)
- LINES (BY TYPE, MAGNITUDE & DEGREE)

RELATIONS AMONG GROUPS
- PATH LENGTH, DISTANCE AND CONNECTION

OVERALL CONNECTIVITY OF THE STRUCTURE

Figure 4. Dimensions for the characterization of ASK structures.

The definitions of all of the terms and characteristics used in our scheme are listed in the Appendix. Our method was to go through all of our ASK representations, and to characterise and classify them by this scheme. This gave us some way to describe the representations in purely structural terms.

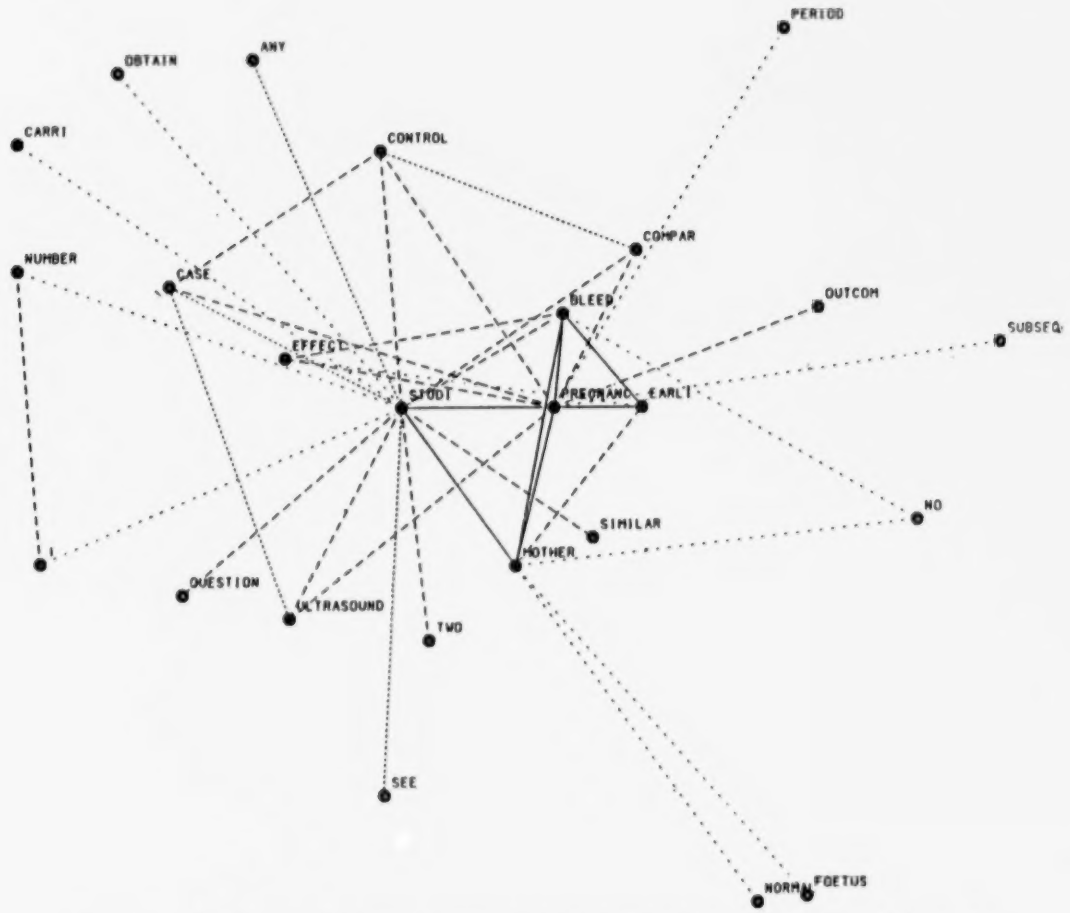


Figure 3. ASK representation for s.14, derived from the problem statement of Figure 2.

2.4 ASK - Text relations

To to discover groupings of ASKs which lead to choice of retrieval strategy, we considered the relationships between ASK structures and the structures of texts which were evaluated in respect of those structures. Since we had no a priori schema, this part of the study consisted of a highly exploratory and informal data analysis, based on the usefulness evaluations and comments of the subjects, and on visual inspection of the structures representing texts and ASKs. This aspect of the study resulted in a specification of a retrieval strategy for each of the ASKs, which would have resulted in ranking the evaluated documents in the order of their usefulness (or in not retrieving the not useful documents).

In this portion of the data analysis it became evident that some lexical information would be required, in addition to structural, in order to choose appropriate retrieval strategies. For example, terms such as 'RESEARCH', 'WANT', 'FIND' and 'PROBLEM' usually indicated areas of the ASK structure which were substantive to the topic of search, whereas terms such as 'LITERATURE', 'TODAY' and 'SEARCH' were associated with areas of the graph concerned with output characteristics. This led us to develop several closed vocabulary sets for identifying areas of the ASK graphs which could be used for different aspects of retrieval strategy formulation.

Thus, the candidate strategies that we developed for each ASK depended on identifying particular areas and substructures of the ASK graph which would allow identification of particular structures of specific lexical items in the representations of potentially useful texts, and provide some means of ranking. These areas and substructures were found, at this stage of analysis, by quasi-algorithmic techniques, which were associated in each case with the general structural characteristics of the ASK representation already assigned.

Figures 5 - 9 are representations of abstracts of documents which were judged, respectively, very useful, quite useful, marginally useful and not useful to the ASK represented in figure 3. As an example of our method for arriving at our eventual strategies, and of how the ASK structures were characterized, we reproduce the reasoning we used in this case.

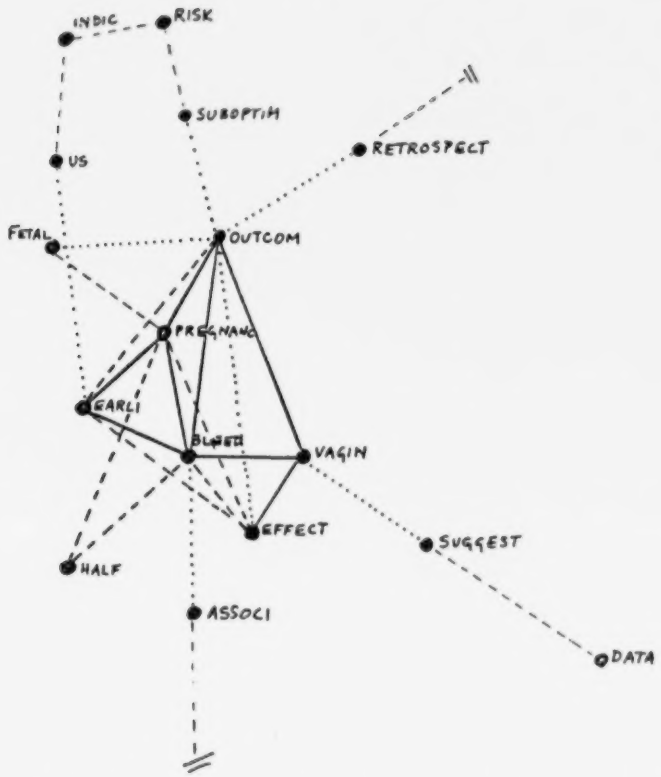


Figure 5. Representation of document 14.01 (judged very useful).

The ASK structure for S.14 is characterized as indicated in figure 10. From the structures of the five evaluated documents for this subject, it is evident that the basic strategy must be to look for documents which center on the level 1 nodes in the type 1 cluster, but that this strategy alone, as simple matching, would not account for the particular ranking given these texts. For instance, it appeared that some concepts, such as 'OUTCOM', which were not in the type 1 cluster, were significant. Also, as can be seen from figure 9, the location and associative structure of matched terms in the text representation is as important as the matching itself. We notice, for instance, that the Type 1 cluster of the ASK has several triadic substructures at level 1, all based on the highest degree node in that cluster, 'PREGNANC', and that these characteristics appeared to bear on the usefulness judgements of the texts.

Thus, for this ASK structure, we hypothesize that the highest degree node at level 1 in the ASK, which we take to be some indication of 'centrality', should also be fairly central in the text representation (relatively high degree at levels 1 and 2). Furthermore, text structures which exhibit the same triadic structure as the ASK structure should be ranked higher than those which do not. In conjunction with the latter hypothesis, the triads can be rank ordered according to the sum of their sides. Therefore, preference will be given to a text with the triad 'PREGNANC - BLEED - EARLI' over one with the triad 'PREGNANC - STUDI - MOTHER'. That is, the smaller the circumference of a matching triad in a text, the higher the weight for that text. A further criterion for usefulness appears to be incorporation into the center of the text structure of peripheral nodes from the star based on the most involved type 1 node (in s.14, these are 'OUTCOM', 'SUBSEQU' and 'PERIOD', radiating from 'PREGNANC'). This ranking rule, on the basis of the structures and evaluations, is somewhat weaker than the others. And as the weakest criterion, incorporation of level 2 nodes of the Type 1 cluster into the central cluster of the text structure (i.e. 'ULTRASOUND', 'COMPAR', 'CONTROL' and 'EFFECT') seems reasonable.

Thus, one possible retrieval strategy and ranking mechanism based on these hypotheses for this ASK structure type is:

1. Quorum search on the set of terms
   s = {type 1 cluster nodes; peripheral nodes of the highest degree level 1 star}
   --must contain at least highest degree level 1 node and one other from type 1 cluster.
2. From retrieved set, eliminate any in which highest degree type 1 problem statement node is not at level 1. For remainder, rank according to relative degree 2 of highest degree type 1 node, all documents with equal first or better, ranked 1.

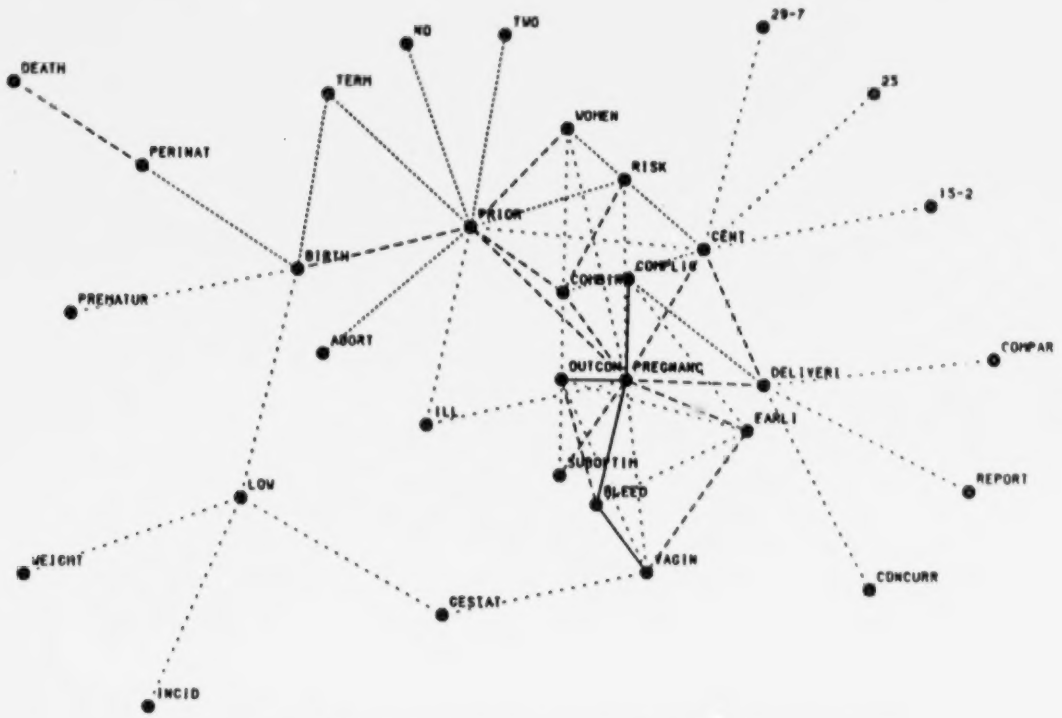


Figure 6. Representation of document 14.05 (judged very useful).

3. Rank within groups determined in step 2 according to triad matching, as follows:

T1 + T2	Both complete at level 1
T1 + T2	One complete, one partial
T1	Complete at level 1
T1 + T2	Both partial at level 1
T1	Partial at level 1
T2	Partial at level 1
T1 or T2	Match at < level 1

where T1 is the smallest circumference triad, T2 the second.

4. Rank within groups determined in step 3 according to star integration, by number and degree of star nodes.

This strategy would, in our example, step-by-step: 1. retrieve all 5 documents; 2. eliminate 14.06, group 14.04, 14.05 and 14.01 in the first rank, and rank 14.03 after all three; 3. rank 14.01 first, 14.05 second and 14.04 third, with 14.03 still fourth; 4. increase 14.01's ranking overall, 14.05's ranking relative to 14.04 and 14.04's relative to 14.03.

Our general method was to go through each ASK-texts set in this manner, using the results gained with each analysis to guide subsequent ones. We followed up by reanalysing the entire set of data, in order to make use of the later results on those sets analysed first. This resulted in a number of specific strategies associated with specific ASK structures.

We then grouped the strategies according to their general characteristics, such as method for choice of terms for initial matching, method for choosing structures for matching, and discrimination or ranking methods. The final step in the study was to identify common characteristics among the ASK structures associated with the groups of retrieval strategies. These last two stages were interactive and iterative.

3. RESULTS

3.1 Data and response rate

We elicited 53 usable problem statements with topics ranging from education and psychology to chemistry and medicine, and users from beginning masters degree students to completing Ph.D. students to M.D.s to professors and independent researchers. Of this group, 40 returned questionnaires, 5 of which had no evaluated documents, or were otherwise unusable. Thus, our problem statement corpus for general categorization is 53, but that for comparison of ASK and text structures is 35. For these 35 problem statements, 298 documents were evaluated, ranging from 2 to 15 per problem statement. We were unable to find abstracts for some of these documents, which brought the final number of documents used for strategy generation to about 250.

3.2 Classes of retrieval strategies

The retrieval strategies for each problem statement were quite complex, as can be seen from the example of s.14. However, they all followed a general two-stage pattern. First, a set of word stems in the ASK structure would be identified, on the basis of structural and lexical features of the ASK, which would be used to retrieve a set of documents by a simple quorum search. Then, this retrieved set would be massaged, with documents

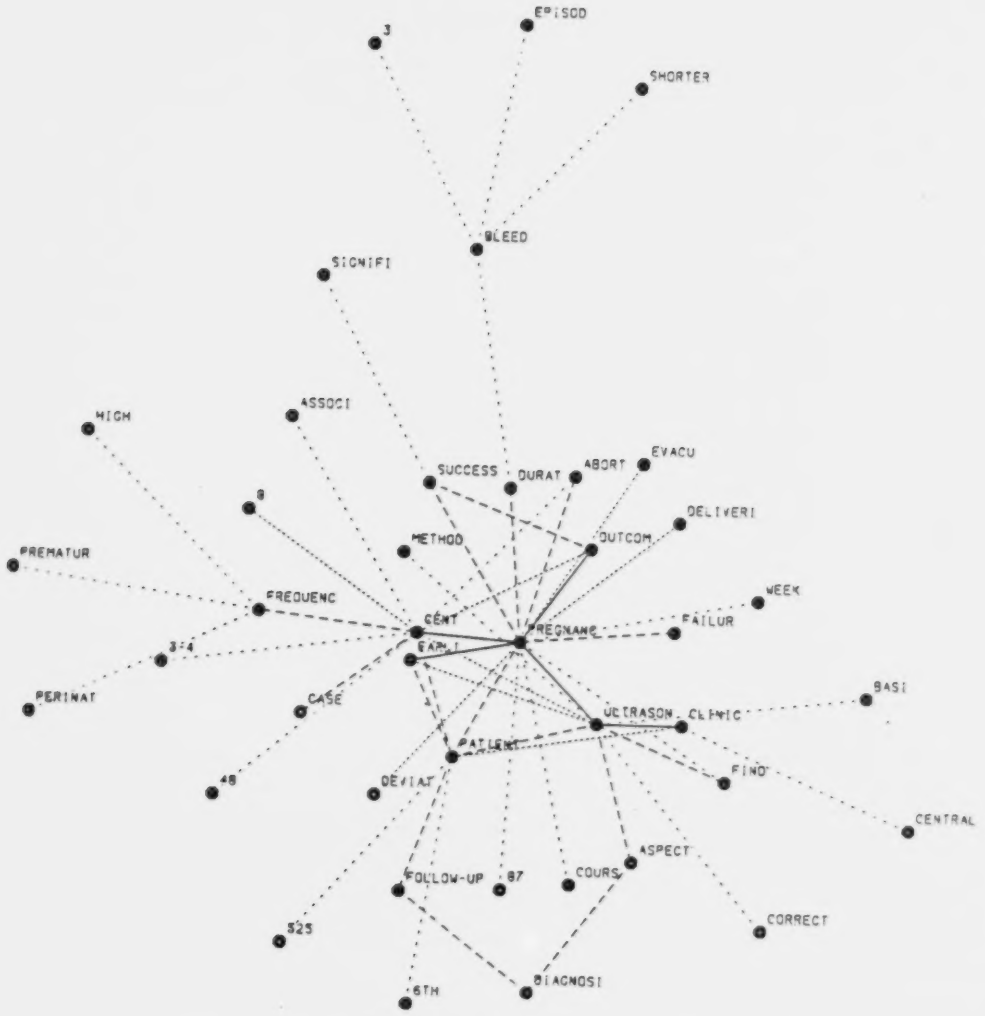


Figure 7. Representation of document 14.04 (judged quite useful).

either discarded or ranked also according to rules derived from the structures and lexical features of the ASKs, which are applied to the structures of the texts.

The complexity of the rules in both stages was, in general, the result of combinations of five different kinds of basic retrieval strategies, which we have labelled MATCH, TRIAD, STAR, PATH, and LEXICAL. The first is simple term identification, the next three are structural in nature, and the last combines with the others by taking account of special closed vocabularies.

We have decided not to attempt an enumerative classifiction of retrieval strategies, but rather to describe the individual basic strategies, which are invoked under specific conditions of ASK structures. Thus, we have a synthetic, faceted classification for retrieval strategies.

These strategies are briefly characterized below.

MATCH	specifies an ASK structure, or area of an ASK structure, from which a list of terms is to be used for straightforward quorum searching.
TRIAD	operates on clusters in the ASK structure, specifiying triplets of terms whose relationships and position in the ASK structure will be used to rank the texts.
STAR	identifies terms for matching and ranking from stars in the ASK structure.
PATH	identifies groups of terms for matching and ranking which are attached to clusters in the ASK structure, but are not parts of clusters. Group relations are retained for ranking purposes.
LEXICAL	identifies 'pointer' or 'non-content' words in the ASK structure, which are eliminated from searching consideration and used to identify specific parts of the structure to be operated upon.

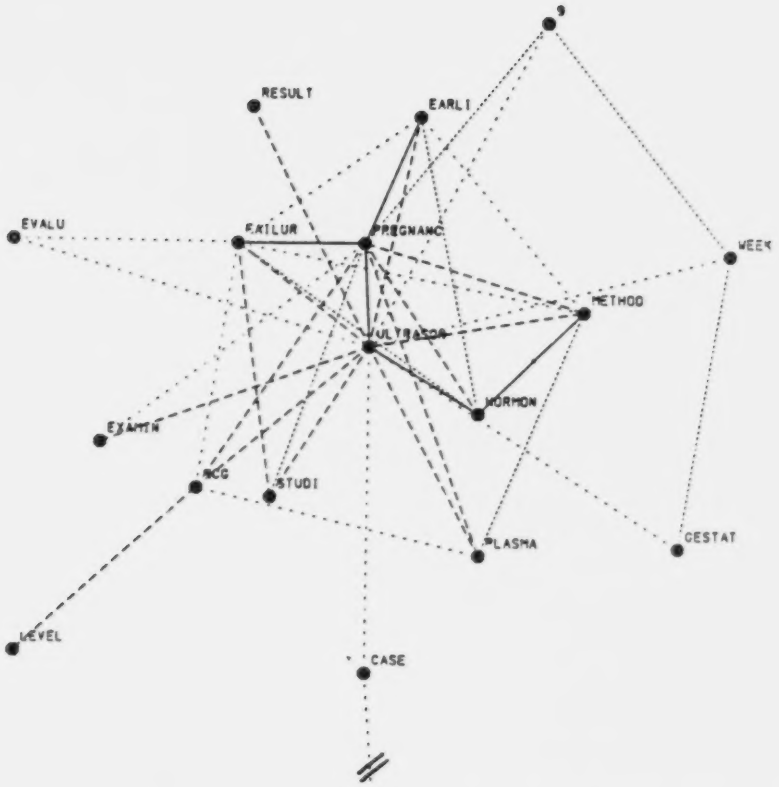


Figure 8. Representation of document 14.03 (judged marginally useful).

Thus, the strategy for s.14 can be summarized as:

STAGE 1
- LEXICAL (finding one closed vocabulary term, _____)
- TRIAD (operating on type 1 and 2 clusters)
- STAR (operating on stars)
- MATCH (using terms from TRIAD and STAR)

STAGE 2
- MATCH (must have most involved node)
- TRIAD (rank in order of structure duplication and node strength)
- STAR (modify rank by inclusion of star nodes).

The rules for invoking the strategies depend upon the structures of the ASKs.

3.3 ASK structures and retrieval strategies

Given the nature of the retrieval strategies we identified, it is obviously more appropriate to identify significant characteristics of ASK structures for strategy invocation, than to attempt an explicit classification. We have identified a number of basic facets of the ASK structures which were regularly connected with the invocation of specific retrieval strategies. These can be viewed as facets in the traditional classificatory sense, with a specific citation order. In this case, we can consider our schema as a synthetic classification which implies a specific order and type of strategy implementation. But given that the purpose of the categorization is to get to retrieval strategies, it might be more clear to view the facets as data-driven rules, applied in a hierarchical manner to specify particular strategies.

Viewed in this way, we found three basic facets (rules associated with one another according to specific criteria). These are called ATTACHMENT, OVERALL STRUCTURE, and STRUCTURE CHARACTERISTICS.

ATTACHMENT is concerned with whether there are two or more structures in the ASK structure which are not connected at all with any of the others, in which case the ASK is termed 'detached'. The OVERALL STRUCTURE facet is concerned with the type, number and connection of clusters in the ASK structure. And the STRUCTURE CHARACTERISTICS facet is concerned with the local structural and lexical features of the ASK, and its overall connectivity. All of these are briefly specified in figure 11.

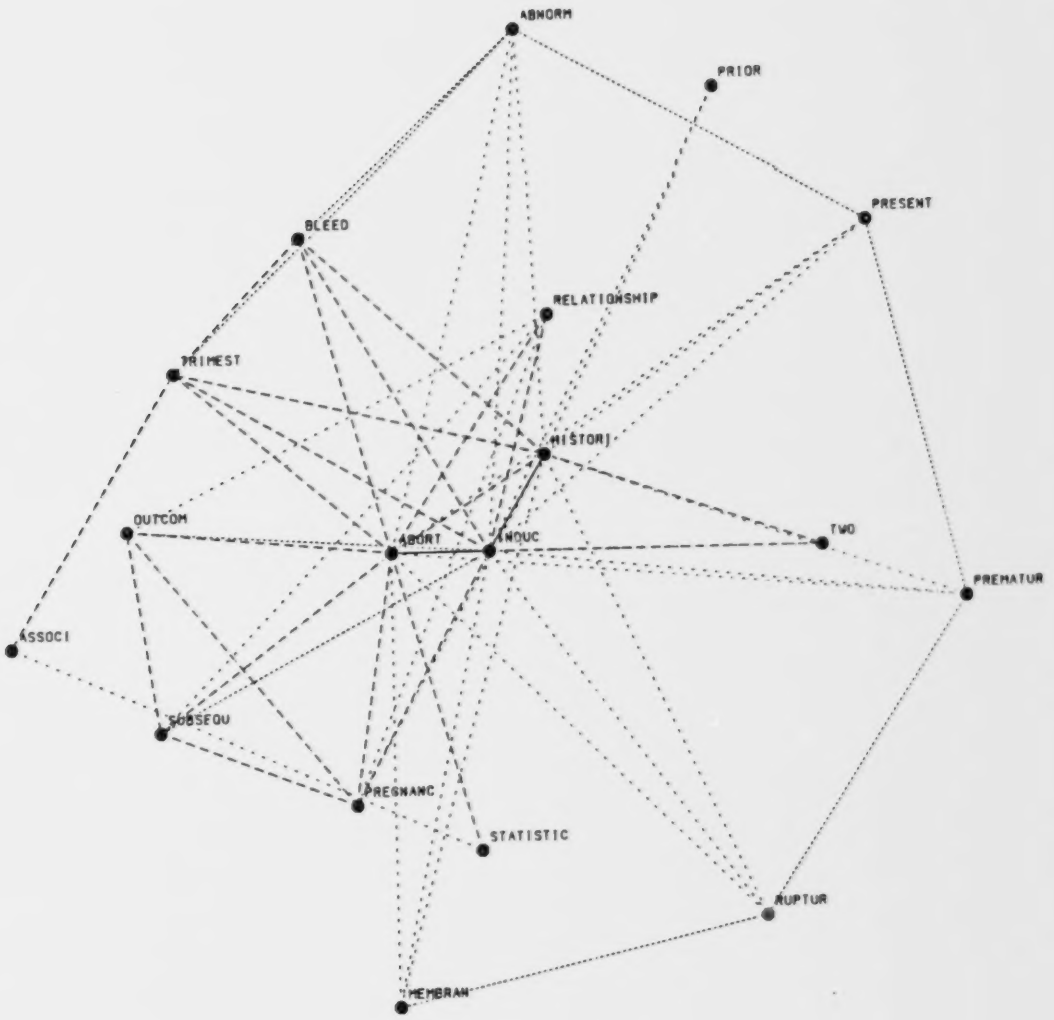


Figure 9. Representation of document 14.06 (judged not useful).

ATTACHMENT
1. Attached
2. Detached

OVERALL STRUCTURE
1. Single type 1 cluster [optimally with incorporated type 2 cluster(s)]
2. Two or more clusters linked at PL 0
3. Two or more clusters with PL > 1
4. No clusters

STRUCTURE CHARACTERISTICS
1. Substantive lexical items in cluster and magnitude of cluster
2. Connectivity of cluster at levels 1 and 2
3. Structure of cluster at levels 1 or 2
4. Number of stars
5. Number of lines

Figure 11. Facets of ASK structures.

The rules for invoking retrieval strategies follow the general form:

If ASK is of category x,
Then do y

where y is either specifying a retrieval strategy or invoking another rule. In order to show how this schema works, we once again return to the example of s.14.

The first facet invoked is ATTACHMENT. The basic rule in attachment says

if attached,
then do OVERALL STRUCTURE.

Since this is not a detached structure, we proceed to the facet OVERALL STRUCTURE. In this facet, s.14 responds to the the rule

if two or more clusters linked at PL0,
then do STRUCTURAL CHARACTERISTICS 1 (label ASK as B2).

STRUCTURAL CHARACTERISTICS 1 is a lexical characteristics rule, which goes:

1. Mark closed set words in clusters
2. If substantive words in type 1 cluster > 2,
   then TRIAD in type 1 cluster
3. If substantive words in type 2 cluster > 2,
   then TRIAD in type 2 cluster
4. If more clusters,
   then 3,
   else do connectivity.

The connectivity rule operative here is:

If high degree level 1 node,
then MATCH on node,
do connectivity.

The connectivity rule that applies is:

If highest degree level 1 node is star,
then STAR.

This will exhaust the possibilities for this particular structure, so that all of the terms identified by the invoked strategies will then be passed to MATCH, for the stage 1 quorum search. Then the subseqent ranking will take place, requiring that PREGNANC be in any relevant document (the MATCH invocation), then ranking by TRIAD inclusion and finally reranking by STAR.

Although there are many possible combinations of characteristics available, as it turns out, the number of specific rule results is small, so that the combinations may be collapsed into classes. These classes (still under investigation), determine the eventual retrieval strategy choice.

4. DISCUSSION

4.1 ASK and retrieval strategy classification and implementation

From the results and examples given in sections 3.2 and 3.3, it seems that a relatively small number of basic retrieval strategies can be used in combination to produce a variety of overall strategies and ranking mechanisms. These basic strategies respond not only to the requirements for straightforward matching, but also for those situations where taking account of general structural information and specific term interactions are necessary. Taken in specific orders, they can reflect the individual strategies discovered in the data analysis.

The characteristics used for classifying the ASK structures (or for invoking the retrieval strategies) are also relatively small in number, yet apparently responsive to relevant aspects of the structures as far as choice of effective retrieval strategy is concerned. This is of some interest, since the citation of the facets tends not to group the ASK structures into what one might think intuitively reasonable classes. For instance, overall connectivity appears not to be initially too important, nor are cluster size or numbers of stars. The most relevant criteria appear to be the number of clusters (no matter what type or size) and the internal structures of those clusters. We do not yet have any interpretations of what these groupings mean in terms of the nature of the users' problems, but are willing for the moment to accept retrieval performance as an adequate justification for them.

The implementation of these strategies appears to be possible if not exactly easy. By performing an initial quorum search, we eliminate the necessity of large-scale structure searching, a difficult process which is thereby restricted to a relatively small subset of documents which can be manipulated locally. Identifying the appropriate structures within the ASK seems likely not to present a problem. Furthermore, there are several natural formalisms for representing our facets and rules, such as frames and productions, which makes us think that this type of retrieval might be implementable in at least a test environment.

CROF86 has recently proposed an interesting scheme for taking account of term dependencies in a probabilistic retrieval environment. It might be of some interest to use problem statements and the structure identification rules proposed here as input to that retrieval mechanism. The ASK structures certainly provide a different rationale for term dependencies than normal frequency data.

A. CLUSTERS (2)

Cluster a	Cluster b
Type = 1	Type = 2
Mag = 9	Mag = 3
Con1 = 7/36	Con2 = 3/3
Con2 = 18/36	
Con3 = 19/36	
Con4 = 20/36	

B. STARS (3)

Star c	Star d	Star e
Type = 1	Type = 1	Type = 1
Mag = 4	Mag = 8	Mag = 3
Deg2 = 1	Deg2 = 3	Deg4 = 2
Deg3 = 1	Deg3 = 5	
Deg4 = 3	Deg4 = 7	

C. LINES (0)

D. RELATIONS

a - b	a - c	a - d
PL = 0 D = 0	PL = 0 D = 0	PL = 0 D = 0
Con1 = 5/27	Con1 = 5/9	Con1 = 3/9
Con2 = 14/27	Con2 = 9/9	Con2 = 7/9
Con3 = 17/27		Con3,4 = 7/9
Con4 = 17/27		

a - e
PL = 0 D = 0
Con1 = 4/9
Con2 = 5/9
Con3,4 = 5/9

b - c	b - d	b - e
PL = 0 D = 0	PL = 1 D = 1	PL = 1 D = 1
Con1 = 1/3	Con1 = 1/3	Con1 = 1/3
Con2 = 3/3	Con2 = 2/3	Con2,3,4 = 1/3
	Con3 = 3/3	

E. OVERALL CONNECTIVITY

n = 25 l_{max} = 300

7	Con1 = 7/300	5	Con3 = 30/300
18	Con2 = 25/300	11	Con4 = 41/300 = 0.13667

Figure 10. Characterization of ASK structure s.14

5. CONCLUSION

5.1 Retrieval strategies and ASKs

Even on the basis of the highly preliminary results presented here, it appears that it is possible to use characteristics of ASK representations to specify different retrieval strategies which are responsive tothe users' ASKs. The facets identified as useful in this study do group ASK representations in ways which seem to distinguish them one from another and also to imply appropriate, and substantially different retrieval strategies. The rules for identifying the ASK structures, and for implementing the retrieval strategies, seem within the capabilities of even present IR system implementations (given a suitable front-end). Thus, there is now some hope for answering the questions posed at the beginning of this paper. Nevertheless, our results are only indicative, and will require implementation and evaluation in a real test environment. This will be the subject of a further study, perhaps making use of CROF86's results.

5.2 ASK representation and human-computer interaction

The ASK project began with a design study initiated in 1978. Although various aspects of that original design have changed through the course of the project, two have remained firm: the basic ASK hypothesis, that people should not be forced to specify their information 'needs'; and, the narrative monologue problem structure. The validity of the former is, we believe, if anything strengthened by the results of this study, but we feel that it may be appropriate now to modify the latter.

We make this suggestion for several reasons. First, we wish to take account of results from studies by ourselves and others [BELK83; BROO85; CROF85], which stress the importance of interaction between user and intermediary in the building up of the intermediary's model of the user. One important aspect of that model is the model of the user's problem [BROO86; CROF85] or state of knowledge; that is, of the user's ASK.

Second, in our ASK projects, we have attempted to capture sufficient linguistic data in the initial problem statement, so that that statement alone could provide the basis for an adequate ASK representation. This has meant long narratives, with very few interventions by the experimenters. Although we tend not to worry about hardware, or even software constraints on our general system design, it seems that we should perhaps not count on speech understanding systems of the complexity required for this sort of data in the too near-term future.

Finally, our results indicate that a progressive building up of an ASK structure, via graphic interaction by the user with the intermediary's model of the ASK, might be more effective and efficient in developing accurate ASK representations, and in identifying important aspects of the ASK, than a one-time monologue.

The ASK classification and retrieval strategy specification will be valid whether the ASK structure is arrived at in a one-time or progressive manner. Indeed, it appears likely that our results could be used to guide progressive ASK representation. Therefore, for the reasons specified above, and in particular in order to integrate the results of this project into the distributed expert model of information retrieval, whether as an 'intelligent information provision mechanism' [BROO85] or 'expert assistant for document retrieval' [CROF85], we suggest that the next step in ASK investigation should be embedding ASK construction in an interactive dialogue between user and computer.

Our problem statement elicitation could, indeed, stand as a basis from which to begin such investigation, since its tripartite structure corresponds rather well to several opening and subsequent gambits often used by human intermediaries in information interaction [BROO83]. This type of interaction also coincides well with suggestions for driving such human-computer dialogues [DANI85]. And such a progressive building up of the ASK structure appears to match well with [CROF85]'s suggestions for a Request Model Builder. We are encouraged, therefore, that our results in this project, suggesting ways of distinguishing IR system user situations in ways which are directly useful for determining retrieval strategies, do not stand alone, but rather support and offer insights for other work on intelligent information systems.

Acknowledgements

The data collection for this project was organized and conducted by Kevin Hapeshi, who was also responsible for the associated programming and many developments in the text analysis and representation. We wish to thank the staffs of the Central Information Services and the library of the Institute of Education, both of the University of London, for allowing the data to be collected from their patrons, and for co-operating so whole-heartedly in this study. Our greatest thanks go to the users, from whom we asked so much, and who responded so generously.

REFERENCES

BELK80 BELKIN, N.J. (1980). Anomalous states of knowledge as a basis for information retrieval. Canadian Journal of Information Science v.5: 133-143.

BELK82 BELKIN, N.J., ODDY, R.N. & BROOKS, H.M. (1982). ASK for information retrieval: Parts I & II. Journal of Documentation 38: 61-71, 145-164.

BELK83 BELKIN, N.J., SEEGER, T. & WERSIG, G. (1983). Distributed expert problem treatment as a model for information system analysis and design. Journal of Information Science v.5: 153-167.

BROO86 BROOKS, H.M. (1986). Developing and using problem descriptions. Paper presented at IRFIS 6: Intelligent Information Systems for the Information Society, Frascati, September, 1985. Amsterdam, North-Holland, in press.

BROO83 BROOKS, H.M. & BELKIN, N.J. (1983). Using discourse analysis for the design of information retrieval interaction mechanisms. Proceedings of the ACM SIGIR Conference, Washington, D.C., 1983. SIGIR Forum 17:31-47.

BROO85 BROOKS, H.M., DANIELS, P.J. & BELKIN, N.J. (1985). Problem descriptions and user models: developing an intelligent interface for document retrieval systems. In: Advances in intelligent retrieval. Proceedings of Informatics 8. London, Aslib, in press.

CROF84 CROFT, W.B. & THOMPSON, R.H. (1984). The use of adaptive mechanisms for selection of search strategies in document retrieval systems. In: Research and Development in Information Retrieval. Proceedings of the Third Joint BCS and ACM Symposium, Cambridge, 1984. Cambridge, Cambridge University Press: 95-110.

CROF85 CROFT, W.B. & THOMPSON, R.H. (1985). An expert assistant for document retrieval. COINS Technical Report 85-05. Amherst, Mass., Department of Computer & Information Science, University of Massachusetts.

CROF86 CROFT, W.B. (1986). Boolean queries and term dependencies in probabilistic retrieval models. Journal of the American Society for Information Science 37:71-77.

DANI85 DANIELS, P.J., BROOKS, H.M. & BELKIN, N.J. (1985). Using problem structures for driving human-computer dialogues. In: RIAO 85. Actes of the conference: Recherche d'Informations Assistée par Ordinateur, Grenoble, March 1985. Grenoble, I.M.A.G.: 645-660.

HAPE85 HAPESHI, K. & BELKIN, N.J. (1985). Developing and evaluating an au on-line information retrieval system based on user problem statements. In: RIAO '85. Actes of the conference: Recherche d'Informations Assisté par Ordinateur, Grenoble, 1985. Grenoble, IMAG:681-698.

ODDY77 ODDY, R.N. (1977). Information retrieval through man-machine dialogue. Journal of Documentation 33:1-14.

PALM84 PALMQUIST, R.A. & EISENBERG, M. (1984). Testing a text analysis metric using magnitude estimation. In: Proceedings of the ASIS Annual Meeting 21:231-236.

PORT80 PORTER, M.F. (1980). An algorithm for suffix stripping. In: New models in probabilistic information retrieval. C.J. van Rijsbergen, S.E. Robertson and M.F. Porter. BLROD Report 5587. Cambridge, Computer Laboratory, University of Cambridge: 98-106.

WEST83 WESTLAND, A. (1983). Text analysis for an ASK-based information retrieval system. M.Sc. Dissertation, Department of Information Science, The City University, London.

APPENDIX

GRAPH CHARACTERISTICS

We are characterizing our problem statement graphs according to the following features:

NODES and LINKS

The **DEGREE** of a node is the number of links incident on that node.
The **LEVEL** of a link is the association strength category of the link.
The **LEVEL** of a node is the maximum link level incident on that node.

GROUPS

A **GROUP** is a CLUSTER, STAR or LINE.
The **MAGNITUDE** of a group is the number of nodes in that group.
The **PATH LENGTH** between two groups is the minimum number of links that must be traversed to get from a node in one group to a node in the other group. The PATH LENGTH between two groups with a common node is 0. A PATH LENGTH of 1 is a **DIRECT** path.
The **DISTANCE** between two groups is the maximum link level connecting any two nodes, one in each group. For groups with shared nodes, DISTANCE = 0. Otherwise, DISTANCE applies only to DIRECT paths.
The **CONNECTION** value between two groups is the ratio of actual links between nodes in the two groups to the maximum possible links between them. CONNECTION applies only to DIRECT paths. CONNECTION at level 2 is the ratio of level 2 links to maximum, at level 3 of level 2 + level 3, at level 4 of all links. CONNECTION applies only to cluster-cluster, cluster-star and cluster-line paths. Maximum values for each are, respectively n x m, n and n links (where n and m are the number of nodes in each cluster).

CLUSTERS

CLUSTERS are of two TYPES.
TYPE I CLUSTER: a set of LEVEL 1 nodes which can all be reached directly by traversing level 1 links, and any level 2 nodes connected to any of the level 1 nodes in the cluster by at least two level 2 links.
TYPE II CLUSTER: a set of nodes of at least level 2 which are connected by at least two level 2 links, but not level 1 links, to other nodes in the cluster.

The **CONNECTIVITY** of a cluster is the ratio of number of links in a cluster to the maximum number of links for the number of nodes in the cluster (l_{max}). Connectivity at level 1 is the ratio of level 1 links to l_{max}, at level 2 of level 1 + level 2 links, at level 3 of level 1, 2 and 3 links, at level 4 of all links.

$$l_{max} = \frac{n(n-1)}{2}$$

where n = number of nodes in cluster.

STARS

A STAR is a set of nodes with one node (the CENTRAL node) connected to at least two nodes of degree 1.
The TYPE of a star is the level of the central node.
The DEGREE of a star is the number of links incident on the central node. Degree at each level is the number of links incident on the central node at that, and all higher, levels.

LINES

A LINE is a set of nodes with the pattern:

degree 1 - [degree 2]^{n}, where n>1, and indicates repetition.

The TYPE of a line is the number of links in that line. Degree at each level is the number of links at that, and all higher, levels.

OVERALL CONNECTIVITY

The OVERALL CONNECTIVITY of a graph is the ratio of number of links in the graph to the maximum number of links possible (l_{max}) for the number of nodes (n). OVERALL CONNECTIVITY at each level is the ratio of the number of links at that, and all higher levels, to 1

PROBLEM STATEMENT ANALYSIS

Each problem statement graph is characterized as follows:

*GROUPS

A. CLUSTERS (total number)
- TYPE
- MAGNITUTUDE
- CONNECTIVITY (BY LEVELS)

B. STARS (total number)
- TYPE
- MAGNITUDE
- DEGREE (BY LEVELS)

C. LINES (total number)
- TYPE
- MAGNITUDE
- DEGREE (BY LEVELS)

RELATIONS AMONG GROUPS

D. CLUSTER-CLUSTER
- PATH LENGTH
- DISTANCE
- CONNECTION (BY LEVELS)

E. CLUSTER-STAR
- PATH LENGTH
- DISTANCE
- CONNECTION (BY LEVELS)

F. CLUSTER-LINE
- PATH LENGTH
- DISTANCE
- CONNECTION (BY LEVELS)

G. STAR-STAR
- PATH LENGTH
- DISTANCE

H. STAR-LINE
- PATH LENGTH
- DISTANCE

OVERALL CONNECTIVITY (BY LEVELS).

*GROUPS are identified by lower-case letters, in the following sequence:

1. TYPE I CLUSTERS, ordered according to highest association strength within the cluster
2. TYPE II CLUSTERS, ordered as above
3. STARS, ordered according to TYPE.
4. LINES, ordered according to TYPE.

Document Presentation and Query Formulation in Muse

S. Gibbs *
D. Tsichritzis **

* Istituto di Elaborazione della Informazione
Consiglio Nazionale delle Ricerce
Pisa, ITALY

** Centre Universitaire d'Informatique
Universite'de Gene've
Geneva, SWITZERLAND

Abstract

Several problems of document presentation and query formulation arising in systems dealing with multimedia documents are discussed. Examples from a prototype distributed multimedia document filing system are described.

1. Introduction

Recent technological advances have made a new kind of information retrieval system possible - systems which store and retrieve multimedia documents [IEEE85]. These systems are a reply to the demands of applications such as office information systems for more extensive document retrieval capabilities. The distinguishing characteristics of multimedia document filing systems are, first and foremost, the attention given to multimedia data, ie text and image data, and with increasing frequency, audio data. Secondly, since these systems are often intended for office use there is a need for interactive as opposed to offline retrieval, and retrieval of complete documents rather than document citations or abstracts. Finally, "multimedia documents" is really a catch-all phrase encompassing things such as forms and letters, in addition to what are more commonly thought of as documents.

Some of the work reported in this paper was done while the first author was visiting the Institute of Computer Science, Heraklion, Greece. Funding was provided in part by the Commission of the European Communities through ESPRIT Project 28 (A Multimedia Filing System) and in part by the NATO Science for Stability Program.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

There are many unsolved problems in the design of multimedia document filing systems. Since documents in these systems are online, there is a possibility for integration with other software components such as mailers and formatters, this implies that standards for document representation must be developed. Multimedia document filing systems deal with updatable rather than static documents. This influences the system design in such areas as authorization and storage organization. The role of a multimedia document filing system within larger distributed office systems is an open question and raises such issues as system architecture and distributed query processing.

In this paper we look at two practical problems, document presentation and query formulation, in the multimedia context. Several new difficulties arise. For example, multimedia documents can be extremely large, both in terms of storage size and number of pages. The user will want to quickly browse through these documents and will not tolerate long delays in the presentation of a particular page. As another example, multimedia documents are highly structured data objects. Structural information is often encoded in a multimedia document in order to allow processing (editing and formatting) and to guide presentation. It is possible then that a document query may involve document structure. For instance, one might imagine a query such as "find all documents with footnotes". Yet it is doubtful whether such queries are common and so knowledge of footnote structure does not seem essential for query formulation. This raises the question of just what structure is useful in formulating queries.

Earlier work on multimedia documents has been concerned with editing, formatting and mailing [Wild82, IEEE85], and document representation [ISO84, Barb85]. Work has also been done on extending the range of what may loosely be referred to as the data types contained by a multimedia document. For example, systems with documents containing spreadsheets [Fors84], interactive graphics [Fein82], and animated sequences [Weye85] have been developed. Less attention has been given to the retrieval of multimedia documents. In [Gibb83] a data model and query language for general office objects is presented. However the focus of this work was on representation rather than presentation. Document retrieval, mainly in the context of bibliographic systems, has been studied for many years. However the indication from the small number of prototype multimedia document filing systems that have been developed [Tsic83, Bonn84, Chri84, Gibb86] is that the multimedia capability brings with it a new range of problems.

The paper is organized as follows. The next section outlines one particular multimedia document filing system, Muse. In Sections 3 and 4 of this paper we describe more fully document presentation and query formulation in multimedia systems. In each case we describe the inherent problems and, when possible, how these problems are dealt with in Muse. We conclude with suggestions for the development of future multimedia filing systems.

2. Muse Overview

Muse is an experimental multimedia document filing system based on a distributed architecture. A Muse document may contain attribute data (named integer or fixed-length character string values), text data (arbitrary length character strings with embedded font control sequences), image data (raster bitmaps, either color or black/white), graphic data (sequences of graphic primitives, such as vectors or circles), or audio data (digitized utterances).

Muse supports the notion of document types. A document type consists of a number of attributes plus a single component named BODY. Values of attributes are either integers or character strings, values of BODY are mixed text, image, graphic and audio data. Document types are arranged in a hierarchy. Subtypes inherit the attributes of their ancestors. The root type is called GENERIC and contains only the BODY component. Specializations of GENERIC contain BODY plus attributes. For example, a document type named REPORT could include TITLE and PROJECT attributes.

Muse can be divided into two parts: a set of document servers which store documents and process document queries, and a set of clients which issue document queries and display documents. Communication between servers and clients is over a network connection. Both servers and clients may be added or removed without affecting the other servers or clients.

The operations provided by document servers are specified by the Muse server interface which governs the encoding of server requests and replies. Table 1 lists the requests presently supported by Muse servers.

request	reply
INSERT <doc>	: <rc, pdi>
FIND <query>	: <rc, count, <pdi>>
GET <pdi>	: <rc, doc>
DEFINE-TYPE <type-defn>	: <rc>
GET-TYPE <type-name>	: <rc, type-defn>
LIST-TYPE <>	: <rc, count, <type-name>>

Table 1 The Muse Server Interface

The first command adds the document <doc> to a server. The return code, <rc>, indicates whether the command is successful; if the latter, a physical document identifier, <pdi>, is also returned. The FIND command causes <query> to be processed on the server. The reply includes a return code, the number of documents satisfying the query plus their physical document identifiers. The GET command is used to bring a document, identified by <pdi>, to the client. The DEFINE-TYPE, LIST-TYPE, and GET-TYPE commands allow a document type definition to be added to a server, the names of existing types listed, and a particular type definition to be sent to the client.

A Muse client allows a user to interactively generate server requests and display replies. The client software runs on a Sun workstation equipped with a high resolution (1152 x 900) black and white monitor, a medium resolution (640 x 480 x 8) color monitor, and a mouse. A Masscomp MC-500 workstation was used for audio input and output. A lengthier description of Muse clients and servers can be found in [Gibb86].

3. Document Presentation

3.1. General Issues

By document presentation we mean the viewing of a document by a user. There are two general issues connected with presentation: performance and the user interface. Performance is clearly crucial. If multimedia filing systems are to act as substitutes for paper-based systems then it must be possible to browse through electronic documents as quickly as through paper documents. For the same reason the

user interface must be well designed. Here the basic question is how the user perceives documents. There seem to be two choices - a knowledge base oriented approach and an object oriented approach. In the first case documents do not have any permanent existence but are generated from a knowledge base in response to a users query. The document generation operation could take into account such factors as the user's background and preferences. In the second case documents are clearly identifiable objects within the system (although the documents may be linked together to form arbitrary networks).

3.2. Muse Approach

A user's conceptual view of a Muse document is that of a distinct object. The object oriented approach was taken since it mimics the user's view of paper documents and, furthermore, certain document operations, such as mailing, are easier to interpret using this approach. At present Muse documents cannot be linked together, however we plan to add a mechanism for grouping documents into collections.

We will now examine how Muse deals with some of the practical problems associated with document presentation. Specifically, we will consider the four problems of responsiveness, scrolling, structure presentation, and audio presentation.

First consider responsiveness. In Muse a useful measure of this quantity is the time taken from the issuing of a GET request to the display of the first page of the document. (Note, this is not the query processing time.) The strategy followed by Muse in reducing this time interval is:

> The GET parameter, a physical document identifier, refers directly to the storage location of the document on the server.
>
> A document is stored in its entirety sequentially at the location indicated by its physical document identifier.
>
> A document is stored in its transmission format so no conversion is necessary before sending the document from the server to the client.
>
> Similarly the transmission format is the same as the display format so no conversion (such as decompression) is needed on the client.
>
> The server code which reads a document from storage onto the network is a simple read/write loop.
>
> The Muse client keeps track of which documents have been downloaded from the server and will not issue a GET if the document already exists on the client.

The result is that the display of the first page of a moderate sized document (say 10 kbytes) takes only a few seconds, irrespective of the size of the server database. This time could be improved in a number of ways. One method is to use page templates, ie dividing a document page into two parts - a varying part (the foreground) and a constant part (the background). It is only necessary to send the template, or background, once. A second improvement would be to send single pages between the server and client rather than entire documents. However, this was not done in Muse since one of the primary design decisions was that the server interface consist of documents and operations on documents rather than pages and operations on pages.

Transmitting compressed data also improves responsiveness provided the time saved in transmission exceeds the time needed for decompression (and compression if the document is stored uncompressed). In the case of Muse no special hardware was available for decompression (such as expanding run-length encoded images) so it was decided to avoid the need for decompressing data at the expense of a longer transmission time.

We will now look at how scrolling is handled by Muse. Figure 1 shows a document on display at a Muse client. The screen is divided into four main regions or subwindows. The rectangular band extending along the top edge is known as the attribute subwindow. Next to it, in the top righthand corner is the exit subwindow. The large area below the attribute subwindow is referred to as the display subwindow and running along its right edge is the scroll subwindow. It is this last subwindow which controls all scrolling operations. First of all note that this subwindow is fairly large in size. It was felt that since scrolling is such an important activity it should be allocated a significant amount of the screen area. The symbols within the scroll subwindow are mouse selectable - the "T" shaped symbol takes one to the top (first page) of a document, the up arrow takes one to the preceding page, the down area to the next page, and the inverted "T" to the last page of the document. The short horizontal bar and intersecting long vertical bar in the middle of the scroll subwindow form a scale indicating the relative position of the current page being viewed. Pointing to a position along the scale, and pressing a mouse button, causes the page corresponding to that position to be displayed. For example, to display the middle page of the document one clicks the mouse in the middle of the scale.

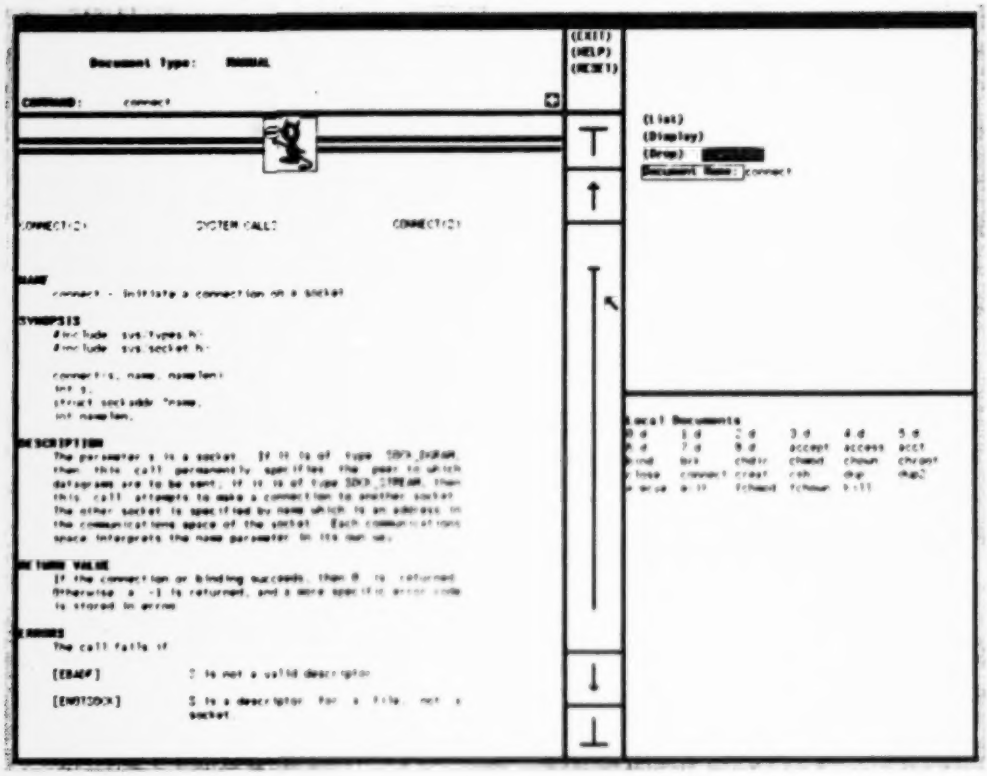


Figure 1 Muse Document Display

This form of scrolling appears adequate for small documents (say ten pages of less) in that one can get to a desired page in a very small number (around three at maximum) of mouse operations. For larger documents a simple improvement would be to allow keyboard entry of a page number. Browsing utilizing document structure [Witt85] is also useful for large documents.

As mentioned in Section 2, Muse documents are classified into document types, each document type has a number of attributes. The type of a document, and the values of its attributes are displayed in the attribute subwindow. The document on display in Figure 1, a rendering of a UNIX* man page, belongs to the document type MANUAL. Figure 1 also shows the single attribute COMMAND with value "connect". If the document type has more than one attribute, the remaining attributes may be viewed by enlarging the attribute subwindow. This is done by hitting the small square appearing in the window's lower righthand corner. (The attribute window is reduced in size in a similar fashion.) The contents of the attribute window do not change as the document is scrolled from page to page.

This approach is workable in Muse since the attribute structure is fairly simple. However in general it still appears useful to provide two "views" of a document - the one consisting of the document content, the other consisting of the document's attributes (sometimes referred to as the profile). Furthermore, certain attributes, such as the title, are very helpful in orienting the user and should be displayed constantly in a fixed location.

A Muse document may contain components that are presented only on explicit request by the user. These components are indicated by icons which appear along the right margin of the document (see Figure 2). There are three types of such icons: those used for audio values (a wiggly line), color images (three empty squares), and black/white images (a solid and empty square). There is no text icon but it would be a simple extension. The component is presented by hitting the icon.

* UNIX is a trademark of Bell Laboratories.

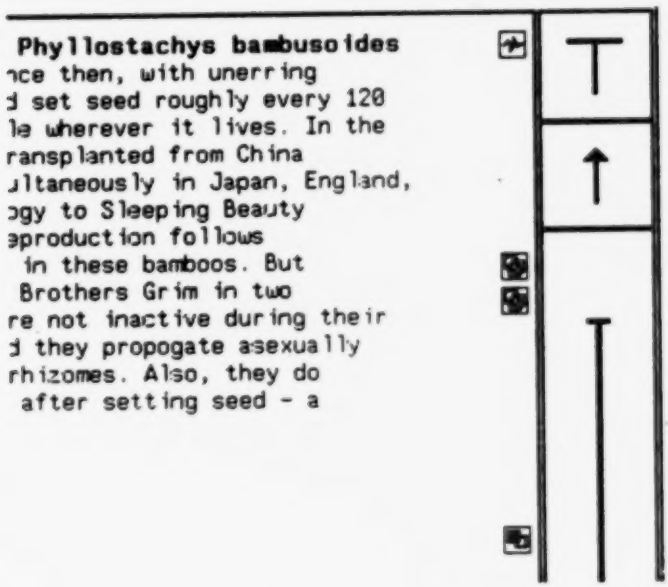

Phyllostachys bambusoides
ıce then, with unerring
ɔ set seed roughly every 120
le wherever it lives. In the
ransplanted from China
ɹltaneously in Japan, England,
ɔgy to Sleeping Beauty
ɐproduction follows
in these bamboos. But
Brothers Grim in two
re not inactive during their
ɔ they propogate asexually
rhizomes. Also, they do
after setting seed - a

Figure 2 Muse Icons

All audio components have this iconic representation so audio values are always presented on the user's, rather than the system's, initiative. This appears adequate for short utterances (up to about half a minute in length). For longer utterances a mechanism is needed, much like the playback on a tape-recorder, for stopping and advancing or rewinding.

Whether an image appears in iconic form is up to the creator of the document. Annotations are a situation where this ability is useful as it allows a viewer to see the document both annotated and unannotated. Furthermore editing and formatting are simplified since the annotations are not interspersed within the text and so can be easily deleted or moved.

4. Query Formulation

4.1. General Issues

In this section we point out some of the differences in the query formulation process when the objects being sought are multimedia documents rather than database records. In general a query is some logical formula stating conditions on the data being retrieved. There are then two factors that determine the types of queries it is possible to evaluate within a system. The first factor concerns the conditions that may appear in a query, the second is concerned with to what these conditions may be applied.

Consider the conditions appearing in queries. In the case of formatted data (ie, database systems) the answer is straightforward - one has the relational operators (=, <, >, and so on) with perhaps additional operators for range checks or substring matches. If repeating groups (multivalued components) occur in the data then one also needs the set membership and subset operators or quantifiers.

If we consider queries on text data the situation is not so clear. One alternative is to forget about the text and just use a multivalued attribute which contains "keywords" extracted from the text. In this case queries have much the same form as above. If the text is used there are various choices as to what may be specified, for example boolean combinations of words or word parts, or regular expressions. In either case one has problems with synonyms and suffixes and prefixes. The traditional information retrieval methods classify documents in terms of concepts they contain. Each of the two basic approaches, full text retrieval and classification, have advantages and disadvantages. From the users perspective full text retrieval (using boolean combinations of words or word parts) is useful when looking for proper nouns or unusual words while classification techniques are more appropriate when searching on a concept.

The situation regarding appropriate query conditions for image or graphic data and audio data is even less clear. One approach is to attach descriptors, much

Figure 3 Muse Query Control

like keywords, to such values. For example an audio value could be tagged with the speaker's name. This would allow such queries as "find letters with audio annotation by Smith". Such queries can be reduced to queries on formatted data and methods for formulating queries on formatted data can be used in these cases. A more sophisticated approach would use a deeper representation of such values (whether audio, image, or graphic) and allow some type of partial matching operation on this representation. However it is not certain how useful such an ability would be in a typical office environment. (Of course, for some specialized applications, such as a picture repository, it would be extremely useful.)

In order to formulate a query the user must identify that part of the document to which query conditions apply. Thus the user must have some knowledge of document structure. Document structure can be described in terms of the logical structure (eg, the relationship of sentences to a paragraph, or a heading to a section) and layout structure (the positioning of information during presentation) [ISO84]. Both these structures are important for editing and formatting but are generally too detailed to be useful in query formulation. For retrieval we believe a third structure, the conceptual structure [Barb85], is more appropriate. The conceptual structure identifies parts of documents that have similar uses, for example the sender's name in a letter or the title and project number in a report. The conceptual structure is much simpler (shallower) than the layout or logical structures and so should be easier for the user to work with when forming a query.

4.2. Muse Approach

Query formulation in Muse begins with the selection of a document type. Two commands are useful at this point: the List Type command lists the names of document types defined on the server to which the client is connected. The Get Type command displays the definition for the type indicated by Type Name (these commands are available in the menu shown in Figure 3).

After selecting a document type, the next step in query specification is entering the query conditions in a "query template". Hitting the Enter Query command enlarges the attribute subwindow and displays the component names for the document type. The user can now select a component, by hitting its name with the mouse, and then enter a condition from the keyboard. Conditions are entered in the form <operator value>. The allowable operators are the relational operators (=, <, <=, >, >=, !=) for attributes (integers and strings) and the contains operator (indicated by '*') for the BODY component. A condition in which the operator is contains, evaluates to true on documents where the value (a list of words or word fragments) appears in the text of the document; the relational operators have the usual interpretation.

As the query is entered in the attribute subwindow a linear form of the query is displayed in the message subwindow (see Figure 3). The linear form resembles a database query - the document type is identified and then followed by the query conditions in the body of a "where" clause.

Once a query has been specified it is issued by selecting Issue. This causes the query to be sent to the server identified by SERVER SCOPE and for the query to be added to a query history maintained by the client for the duration of the session.

The query history is displayed using the History command. Queries are numbered in the order in which they were issued. Each history entry indicates a query number and the number of documents satisfying the query (on the server to which it was issued). It is possible to view a query on the history list using the Go Back command. This causes the query numbered Query No. to be displayed in both the attribute subwindow and message subwindow. By following Go Back with an Edit command it is possible to modify the earlier query.

Both the document structure and query structure used by Muse are fairly simple. In an interactive environment queries tend to be rather simple (as compared to the queries one might find in a program running in a batch environment). However it should still be possible to compose complex queries - an area in which Muse needs some extension. There is a greater need for mechanisms supporting query modification, such as Muse's query history, when dealing with text rather than attribute data. In full text retrieval systems it is often difficult to specify a query which retrieves all documents one is searching for and so mechanisms allowing query modification and experimentation are needed.

5. Future Work

Future work on Muse will be in a number of areas. Considering document presentation and query specification there is a need to build a more flexible user interface and to extend the present query language. Other changes include the addition of new commands for such things as document modification and authorization control. At present Muse documents are constructed in an ad hoc manner, there is an urgent need for a multimedia editor.

The overall architecture of Muse may change in various ways. One likely enhancement will be to add a "type server" which maintains information about document type definitions. In addition a, perhaps separate, "name server" may be added. The function of this server would be to enable potential Muse clients to locate Muser servers. This server could also play a role in such activities as authorization and encryption.

References

[Barb85] Barbic, F. and Rabitti, F., "The type concept in office document retrieval," Proc Eleventh Int. Conf. on Very Large Data Bases (1985).

[Bonn84] Bonnel, R.D., Huhns, M.N., Stephens, L.M., and Mukhopadhyay, U., "MINDS: Multiple intelligent node document servers," IEEE First Int. Conf on Office Automation (Dec 1984).

[Chri84] Christodoulakis, S., Vandenbroek, J., Li, J., Li, T., Wan, S., Wang, Y., Papa, M., and Bertino, E., "Development of a multimedia information system for an office environment," Proc. of the Tenth Int. Conf. on Very Large Data Bases (1984).

[Fein82] Feiner, S., Nagy, S., and van Dam, A., "An experimental system for creating and presenting interactive graphical documents," ACM Trans. Graph. Vol 1 1 (Jan 1982) 59-77.

[Fors84] Forsdick, H.C., Thomas, R.H., Robertson, G.G., and Travers, V.M., "Initial experience with multimedia documents in Diamond," Bull. IEEE Comput. Soc. Tech. Comm. Database Eng. Vol 7 3 (Sept. 1984) 25-42.

[Qibb83] Qibbs, S. and Tsichritzis, D., "A data modelling approach for office information systems," ACM Trans. Office Information Syst. Vol 1 3 (1983) 299-319.

[Qibb86] Qibbs, S., Tsichritzis, D., Fitas, A., Konstantas, D., and Yoergaroudakis, J., "Muse A multimedia filing system," Submitted for publication.

[IEEE85] IEEE., Computer Vol. 18 10 (Oct 1985).

[ISO84] ISO, ISO/TC 97/SC 18/WG 3 N 283 Office Document Architecture.

[Tsic83] Tsichritzis, D., Christodoulakis, S., Economopoulos, P., Faloutsos, C., Lee, A., Lee, D., Vandenbroek, J., and Woo, C., "A multimedia office filing system," Proc. Ninth Int. Conf on Very Large Data Bases (1983).

[Weye85] Weyer, S.A. and Borning, A.H., "A prototype electronic encyclopedia," ACM Trans. Office Information Syst. Vol. 3 1 (Jan. 1985) 63-88.

[Wild82] Wilder, H.A. and Maxemchuk, N.F., "Virtual editing: II. The user interface," Proc. ACM SIGOA Conf. (1982) 41-46.

[Witt85] Witten, I.H. and Bramwell, B., "A system for interactive viewing of structured documents," Commun. ACM Vol. 28 3 (March 1985) 280-288.

An approach to multimedia information management

Gallelli S., Iacobelli C., Marchisio P.
CSELT - Via G. Reiss Romoli, 274 - 10148 Torino, Italy

Abstract

The integrated management of multimedia information, that is of complex information consisting of conventional data, text, graphics, images and voice, is of great interest not only in fields like Information Retrieval, Office Automation, Computer Aided Instruction, Computer Aided Design, but also in other emerging fields such as Tourist Applications, Computer generated films, Newspapers and magazines production, and so on.
In this paper the definition of multimedia object, partially derived from the ECMA standard "Office Document Architecture", is given and an approach to multimedia information management is proposed. A multilevel environment, where multimedia information can be handled, stored and retrieved and the inner level of which consists of a general purpose Multimedia Data Base Management System (MDBMS), is described. They are outlined the main functionalities that languages which describe and manipulate multimedia objects should provide.

1. Introduction

In their activities people create or exchange a lot of information using a wide variety of media. Information can be made of photographs, pictures, films, drawings, speech, music, charts, text, lists of keywords, numbers, characters, and so on: in short every kind of audio-visual information. The integrated management of such complex information, usually named **multimedia information**, is of great interest for a wide range of existing computer disciplines, like Information Retrieval, Office Automation, CAI, CAD, etc., because it allows to extend their use to new application areas such as Tourist applications, Film production, Newspapers and magazines production, etc., and to add benefits to application areas where computers are already used. A more detailed list of possible new application areas has been presented at the 1985 ACM SIGMOD conference [SIGM85].

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.
© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

Information can usually be classified according to the media it belongs to: conventional data, text, graphics, images and voice. Different media imply the solution of highly different problems and several times need different special purpose hardware to satisfy performance requirements. In the past years single media systems have been developed and specific problems related to each media solved (e.g. text retrieval based on signatures, image pattern analysis, voice synthesis, etc.), making the stage set to think of multimedia applications.

Presently many researchers are spending their efforts to solve problems related to the development of multimedia systems. Office Automation is the field where the multimedia problem has firstly been faced and which most efforts have been devoted to [TSIC83, BERT85, RABI85]. Other significant works concern the construction of multimedia workstations [LEDI85, POGG85, BUCC85] and the development of systems able to handle generalized data, like text, voice, images, in a more or less integrated manner [CRAM83, CELL85, NAFF81, LOPE84].

In section 2 the definition of "multimedia object" is given and the organization of a multilevel environment, where multimedia information can easily be handled, stored and retrieved, is shown. In order to solve problems common to a wide range of application domains, the inner level of such an environment consists of a **Multimedia Data Base Management System** (MDBMS). It plays the same role of conventional DBMS in traditional applications.

In section 3 a "multimedia object model", which the MDBMS is based on, is defined. Such a model uses some major concepts coming from the Office Automation world, regarding the organization of documents (e.g. "logical", "layout" and "conceptual" structures) [ECMA85, BARB85], but adapting them to the more complex multimedia environment. Finally the general architecture of the MDBMS we are planning to implement, is outlined and its main components and major functionalities described.

2. Multimedia Information Management

2.1 The Multimedia Object

Although it is recognized that multimedia information consists of text, voice, images and

conventional data, a widely accepted standard definition does not exist yet. In fact, the wide range of applications leads to different approaches in the management of multimedia information [CHRI85, BARB85, CRAM83, NAFF81]. Furthermore, as the field is very young, international standardization is still lacking. Nevertheless most approaches are based on two common concepts. The first is that multimedia information of a system is logically splitted into objects that can often be grouped into classes on the basis of some common properties. The second is that such objects, containing multimedia information, are not "flat objects", but "structured objects", in order to support the greater complexity of multimedia compared with conventional information.

On the basis of the previous considerations, we adopt the following recursive definition of multimedia object:

> "It is the integration of conventional data, non conventional data (text, voice, images, graphics) and other components that are multimedia objects in their turn".

However such definition does not say anything about how such different components take part to the composition of multimedia objects and which is the level of integration from both the "logical" and the "layout" point of view. Logical integration means uniformity in accessing data of different media. Layout integration requires the capability both to gather graphics, images, text and conventional data in a single output area (e.g. in a graphic display) and to syncronize the output of different objects (e.g. an image and its vocal comment).

How such components partecipate and which is their level of integration strongly depends on the specific application. For example an office information system requires a strong integration between text and graphics from both the logical and the layout point of view. On the contrary, in a computer aided instruction system it is important for all the components to be present, even if not strictly integrated (e.g. you could see text and graphics on a display and images, coming from a videotape or a videodisc, on a TV set).

A typical multimedia system deals with highly structured objects with a lot of intricate dependencies and performs very complex operations on such objects. Furthermore, as multimedia information is audio-visual, its target is generally to reach users and seldom to be accessed by other applications for following elaborations. Consequently a multimedia system has to support a heavy user interaction, which requires an efficient and powerful filing-retrieval mechanism and the capability to compose and synchronize information flows. These reasons have led us to extend database techniques to a multimedia environment by defining a Multimedia Data Base Management System (MDBMS). A MDBMS, as well as a DBMS in a traditional context, permits once for ever to solve problems common to a wide range of application domains, relieving the programmer of the burden of actual data management, providing him with a uniform interface, guaranteeing a secure data access, realizing data independence and data sharing, and so on.

2.2 Requirements of Multimedia Data Base Management Systems

A MDBMS can be based on several concepts coming from the conventional database world, although there is a significant set of new requirements related to the presence of new media to be considered.

First of all, a MDBMS must support a more complex **data schema** because logical entities are highly structured objects representing aggregation hierarchies, with much more relationships than in a conventional database environment. Therefore, in order to obtain data independence, a multimedia object should be "self-describing", that is to say stored together with the "structures" containing information about its syntax (format) and its semantics (meaning).

Multimedia reality is generally dynamic, so it must be possible to easily change the database schema during its life, while in conventional databases the schema is known "a priori" and it is necessary to copy all the data out and then reload them to change it.

Another new feature regards how the single media components of multimedia information can be **coded** to be managed, in particular stored, by the MDBMS. Text can be coded like characters in conventional databases (e.g. ASCII code). Graphics is usually coded by storing the syntactic components of a picture (e.g. coordinates of start and end points of a straight line, colour of an area). An image can be coded by a matrix of picture elements (pixels), each of which contains information about hue and intensity; however a high resolution true color image can require up to one million pixels with 24 bit each. Moving images imply an output of about thirty fixed images per second. Voice can be coded by temporal sampling, at least ten thousand samples per second, with about twelve bit each; this is true only for speech, as music requires much more. Therefore managing images and voice is very expensive in terms of both storage and CPU costs; they can also be coded in their cheaper and natural analog form and stored onto separate devices (e.g. videodiscs), but in this case many analysis and processing capabilities are lost. Ample literature on these problems exists: see [STAR85, ANDR84] and references given there.

Conventional data are usually stored into few files in **secondary memory**, consisting of a high number of uniform records, and particular techniques (index, hash) are used to allow a faster access. On the contrary, multimedia data are not formatted and are organized in structures with a great number of files of relatively small size. Even inside the same class there can be objects of very different sizes, so it is difficult to implement techniques that speed up the access.

New features regard **operation types**, too, in order to allow selective data access and powerful insertion mechanisms. In conventional data there are arithmetic, relational and logical operators that act on atomic attributes; insertion is trivial. On the contrary a multimedia environment needs specific algorithms in order to perform content processing and analysis. Some application dependent algorithms can exist, especially for images and voice, and therefore it should be possible to add them to an existing MDBMS. Insertion

regards complex objects with their specific components, so powerful editors are necessary.

Data presentation in conventional DBMS is simple (tabular), while in a MDBMS it can be very complex, so it is useful to store, inside the database, also the way information has to be presented in output (layout structure).

Finally end users can use conventional DBMSs also directly, by means of a **query language**; on the contrary, a MDBMS is only seen as an open system on which a specific application environment is built.

2.3 The multilevel multimedia environment

The multimedia problem can be faced at three different levels, according to the environment level in which multimedia objects are dealt with (see fig. 1).

1. Multimedia object management, independent of both the application and its domain (Multimedia Data Base Management System).

2. General application systems such as Multimedia Office Information System (MOIS), Multimedia Information Retrieval System (MIRS), etc., or systems that use multimedia data and therefore need a MDBMS that manages them in an optimal, integrated, uniform way.

3. Specific applications, built on the top of general application systems (MOIS, MIRS) for a certain class of problems. It is also possible that some applications of a particular field need to use the MDBMS directly, because either there are no general systems for that field or it is convenient for them not to be considered at a general level in order to improve generality, efficiency and compactness of some functions.

In figure 2 it is shown who and how uses the functionalities of the three levels in a complete system environment.

2.3.1 Multimedia data base management system

The multimedia data base management system (MDBMS) is a DBMS oriented to the management (insertion, modification, retrieval) of multimedia objects; it supplies applications of the upper levels with a set of primitives. There are several approaches to the creation of a MDBMS [VLDB83]. A first alternative is to extend a DBMS with new functionalities to manage unformatted data like text, voice, images, graphics in addition to formatted data. Another solution is the cooperation of different systems, each specialized in the management of only one media, through a unique interface. A further solution is the implementation of an "ad hoc" DBMS that manages multimedia objects. There are other possible intermediate solutions taking into account some aspects of the previous alternatives.

Our approach is the design of a new DBMS able to manage multimedia objects in an integrated way using specialized systems for each media. In section 3.2 the architecture of the MDBMS will be shown.

2.3.2 General application systems

The aim of this level is to provide a set of systems specialized in a particular application area (MOIS, MIRS, etc.) and that can be used by more specific applications. A general application system supplies the user (the specific application) with functionalities which allow him to access the information world contained in the Multimedia Data Base using objects familiar to its application area. This can be defined in conformity with possible existing standards, using the

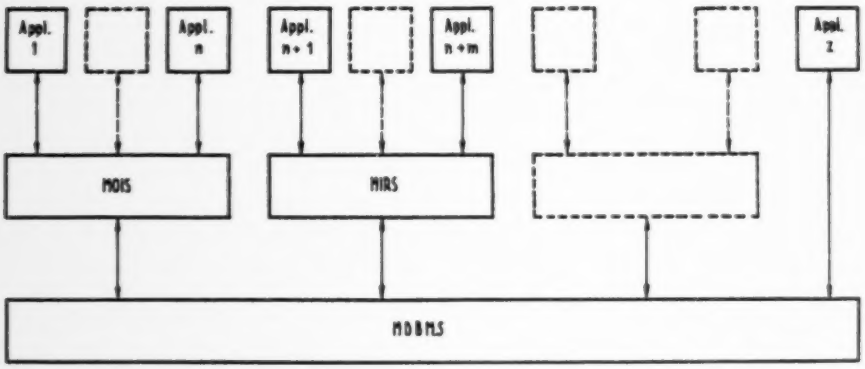


Fig. 1: "Multilevel Multimedia Environment"

MDBMS primitives.

As an example of such an environment, we can consider the **Multimedia Office Information System (MOIS)**. It is a system to manage office information, the objects of which are the various document types circulating in a generic office and defined according to the standard ECMA 101 [ECMA85]. Starting from an office object like the "form", the characteristics of which are the presence of an header, a date, a body, etc., and making use of a primitive like "definition of a specific type of 'form'", it is possible for a specific application of the upper level to define the type "application form", of "form" type, with only three lines in the body, with the date in the upper left corner, etc.

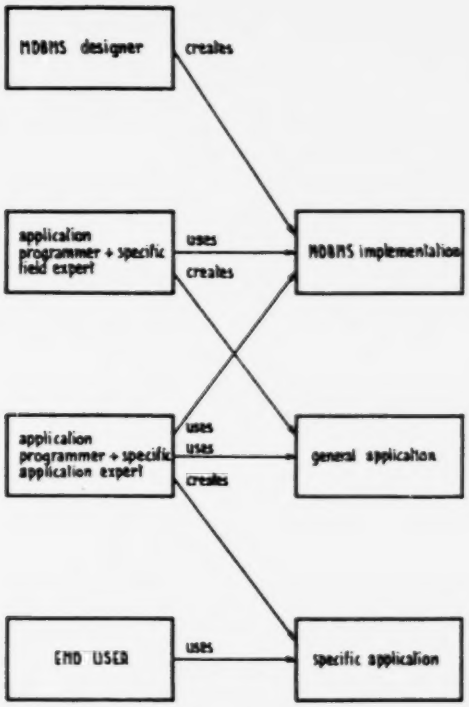


Fig. 2: "Who and how interacts with the components of a multilevel environment"

A wide range of works exists in the literature about office information systems and the standardization of office documents [SMIT84, HORA85, TSIC85, WOO85].

2.3.3 Specific applications

They are the typical applications for end users. They use the functionalities supplied by the general application systems to create other functionalities to support the particular needs of a specific environment. For example, the "document management system", which defines and manipulates the various types of documents circulating in a particular office, will use the functionalities supplied by the MOIS.

Obviously, whenever the necessary functionalities are very particular or not supported by any general application system, those of the MDBMS can be used directly.

Two kinds of languages have been proposed at this level [WOO85, LEDI85], one oriented to "administrators" to define object characteristics, user authorizations, etc., the other to "operators" to perform insertion of instances, retrieval, etc.

3. The Multimedia Data Base Management System

The Multimedia Data Base Management System (MDBMS) constitutes the kernel of any system built to manage multimedia objects. It provides a set of primitive functions which allow the user to create, store, retrieve and manipulate complex objects in an integrated way, that is without having to worry about the possibly different nature of each component of the object itself.

In the following paragraphs the definition of a suitable model to represent an object and of a modular architecture for the MDBMS are examined in detail.

3.1 The Multimedia Object Model

Because of the complexity of a multimedia object, a unique structure may be not powerful enough to represent it; therefore it is convenient to make use of a more sophisticated organization based on a logical, a layout and a conceptual structure. The ideas present in this kind of approach come from the major concepts concerning the organization of documents in the Office Automation world [ECMA85, BARB85], adapted to the more complex multimedia environment.

The **logical structure** describes the "syntax" of the object, namely which components constitute it (e.g. **LETTER** composed of logo, sender's and receiver's addresses, date, text, graphs and signature) and its internal organization (e.g. order of the various components inside the letter, etc.).

The logical structure does not define the way of displaying information. Referring to the previous example, the exact position of each component inside the letter, the size of the logo, the fonts for the text, the scale for the graphs and so on are not fixed yet. Sometimes it may be necessary to specify the temporal sequence of presentation and the lasting of the various components, too, especially if speech or music or moving images are involved. All these parameters are specified in the **layout structure**. As the layout structure of an object generally depends on the layout structures of its components, it is generated taking them into account.

The **conceptual structure** is introduced to explain the meaning of the components of an object; this allows, for example, the retrieval not only by address (or identifier) or by a direct analysis of the content (e.g. understanding the type of a graph), but also by attributes describing either the content itself (e.g. the variables which the graph is function of) or other characteristics independent of it (e.g. what the graph is about).

The conceptual structure can be considered a secondary information level which can be used as a filter while retrieving data, before accessing primary information.

Such structures can be represented with a tree in which the root is the object itself, the other nodes are its components and the edges show the type of correspondence between a father and its children; this correspondence can be either mandatory (at least one occurrence of the child object) or optional (possibly no occurrences), either 1:1 (only one occurrence) or 1:N (several occurrences).

Objects can be grouped into classes on the basis of a set of common properties; so both classes and instances of objects have their own logical, layout and conceptual structures. In order to distinguish between the structures of a class and those of an instance, we refer to the first ones as "generic structures" and to the second ones as "specific structures"; this is an extension of the concept of type and occurrence in conventional DBMSs.

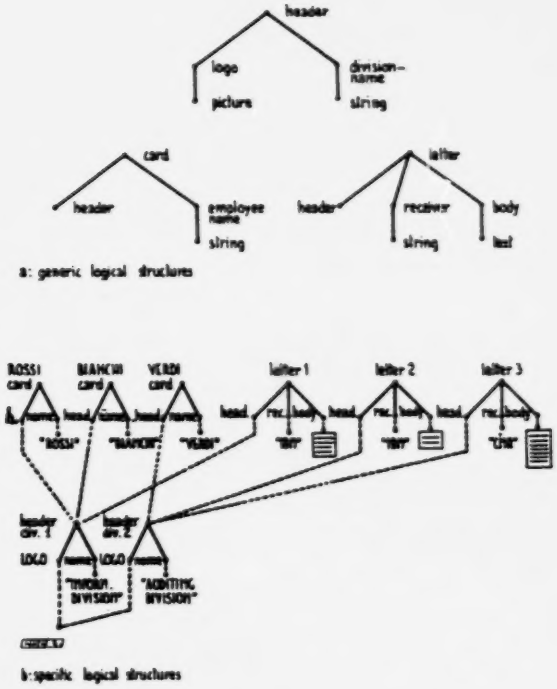


Fig. 3: "Logical structures"

As the components of a multimedia object are generally multimedia objects themselves, they have their own logical, layout and conceptual structures, too; reasoning this way recursively, also objects at the lowest level have their own structures. Moreover every object can be part of one or more other objects while being at the same time perfectly independent. In this circumstance the object is defined only once and then linked to all those which include it. In figure 3 an example of the generic and specific structures of an object ("header") belonging to two other objects ("card" and "letter") is shown.

As a consequence of these considerations, a leaf of the tree representing an object can be either a single media object, namely an elementary component, or a multimedia object, in which case it is logically associated with the tree that actually defines it.

There can be a set of elementary standard objects in which generic logical and layout structures are already completely defined; however it will be possible for some parameters to be specified only in subsequent phases, for example when linking an object to more complex ones or creating instances of it.

Elementary objects will generally be conventional data (integers, reals, character strings, booleans), text, fixed images (e.g. raster), moving images (e.g. coming from videodisc), icons, graphics, audio; in table 1 it is shown how logical and layout structures could be defined for some single media objects.

object	logical structure	layout structure
integer	4 bytes	string up to 10 digits plus the sign
string (N)	array of N bytes	N character long string of a certain font and in a certain direction
boolean	1 bit	"TRUE" or "FALSE"
fixed image	MxN bit map	MxN pixel map on a part of the screen
moving image	MxN bit map sequence	MxN pixel map on a part of the screen for a certain time
graph	array of integers	graph of a certain type on a part of the screen
audio	coded recording	voice output for a certain time

Table 1

In this context associations among objects are solved by creating a new object, the components of which are the objects to be associated. For instance, if you have the object "company" and you want to associate it with the letters sent by that company, you will have to define the object "company-letters" composed of an object "company" and of N objects "letter" (see fig. 4).

3.2 The architecture of the MDBMS

3.2.1 System components

The Multimedia Data Base Management System (MDBMS) is a DBMS with the capability of managing multimedia objects. A traditional DBMS cooperates with some single-media specialized systems to obtain the necessary functionalities.

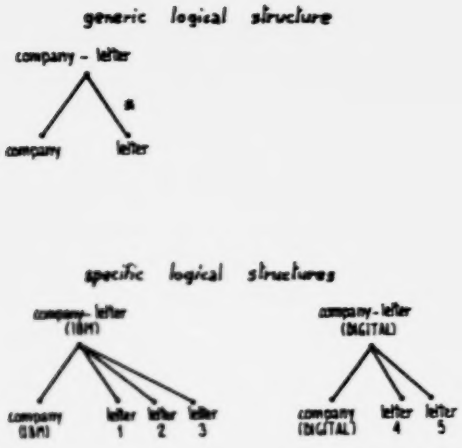


Fig. 4: "Association management"

In figure 5 the architecture of the MDBMS is shown.

The interaction with the MDBMS is accomplished by means of a **Multimedia Data Description Language (MDDL)** and of a **Multimedia Data Manipulation Language (MDML)**, which play the role of the Data Description Language (DDL) and of the Data Manipulation Language (DML) in a conventional DBMS.

The MDDL allows to define, modify and delete the generic structures of multimedia objects.

The definition of a new object can be done following both a top-down and a bottom-up approach; in other terms, it is possible either to specify a certain component and then to define it (this is generally the case when such a component is of no interest outside the context of the object it belongs to) or to define an object and then to use it to build more complex ones.

The Multimedia Data Manipulation Language (MDML) allows to insert, modify and retrieve specific objects.

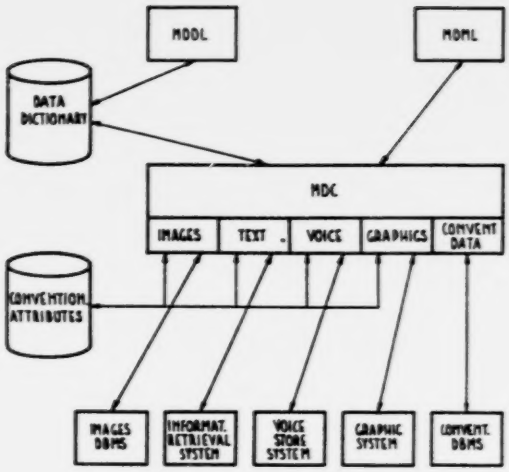


Fig. 5: "MDBMS architecture"

A specific object is an instance of a generic one previously defined by means of the MDDL; it can be generated by defining its specific structures or by modifying those of an already existing one.

All information about the logical, layout and conceptual structures of multimedia objects is stored into a dedicated database, the **Data Dictionary.**

The **Multimedia Data Coordinator (MDC)** is an interface which permits the MDML to uniformly manage all multimedia data stored into the MDBMS. By request of the MDML, the MDC consults the dictionary for what concerns the objects involved in the request and, on the basis of their structures, decomposes the request into several subrequests which have to deal only with single media components. Each subrequest is then submitted to an "ad hoc" single media module which inserts, modifies or retrieves data by means of a specialized system. For instance, a module for graphics could make use of a graphic system and a conventional DBMS to store necessary information, while a module for images should be able to manage them on different devices, considering both images stored as bit maps (onto magnetic disks) and images stored as analogical sequences (onto videodiscs). Finally, when all subrequests are answered, the MDC recomposes the results, generates the answer to the original request and returns it to the MDML.

3.2.2 Major functionalities

Applications that operate on multimedia object structures (generic and specific) can deal with the MDBMS either directly, via the set of

MDDL and MDML primitives, or indirectly, via an "ad hoc" **multimedia editor**, which establishes the proper sequence of calls to such primitives by interaction with the end-user [MEYR82]. For example, in the first case, it is possible to build the logical structure of an object using a "definition" procedure, the parameters of which are the names of its components, while in the second case the editor drives the user by asking him which objects compose the new one and showing him which have already been defined, possibly through user-friendly mechanisms of interaction (e.g. selection from icon-based menus).

Specific objects can be created either explicitly, specifying all their components, or implicitly, for example generating them with special procedures (e.g. graphics) or after retrieving and eventually modifying other objects (e.g. subcomponents, instances, etc.); the two techniques can be intermixed.

During this phase, it might be necessary to switch from the logical to the layout structure and vice-versa, in order to specify the parameters of the generic object that are still undefined (length of text, number of elements of a graphic, etc.).

Due to the complexity of such operations, the use of the multimedia editor appears to be particularly advantageous; referring to the generic structure of an object and driven by its syntax, it can be encharged to ask the user for the right element at the right moment. The editor will be assisted by single media ones when operating on elementary objects.

Modification and deletion of the generic structure of an object may cause inconsistencies with its correspondent specific ones as well as with the structures of those objects which include it or which it includes. Therefore a control mechanism is needed that prevents the user from performing certain kinds of changes, notifying him of inconsistencies that might arise.

The **retrieval** of multimedia objects requires two main capabilities: the first is filtering, that enables the user to specify what he is looking for; the second is browsing, by which he can choose the desired objects among a list of several ones. In a complex query, filtering and browsing should be interleaved: the user, while browsing on previously filtered objects, can modify the selection clauses for a following filtering phase [TSIC83].

There are several methods to perform filtering of specific objects in a multimedia environment.

Two simple ways consist in accessing an object by name (identifier) or in scanning sequentially the objects of a certain class. Such ways, typical of a file system, are weak from the selectivity point of view.

Another method is to perform a search in terms of values of attributes containing conventional data; the selection filter is specified through a query involving a boolean expression concerning such values. This method allows a fast, precise and selectively powerful search, so it is widely used in Information Retrieval Systems, when using keywords, and can be adopted in a MDBMS, too, in order to retrieve an object in terms of its logical structure (e.g. subcomponents, objects which it takes part to, level inside the structure), its layout structure (e.g. position in the output area, dimensions, lasting) and its conceptual structure (e.g. abstractions of its contents). The conceptual structure is particularly important for elementary objects and implies an "a priori" knowledge of the properties which are significant for searching purposes (e.g. names of the things contained in an image, talkers of a speech) in order to define the proper attributes. Especially for images and voice, such conceptual abstractions permit to avoid complex pattern recognition algorithms.

Another approach is to retrieve elementary objects on the basis of patterns present in them. It works quite well for texts, as they can be qualified according to a regular expression of strings present in them; for images and voice, pattern recognition is much harder because of both the noise, which can hide relevant information, and the not clear distinction between patterns.

A further approach is to leave the task of performing pattern recognition to human eyes/ears, by representing objects in terms of abstractions (e.g. miniatures or icons for images, fasttalks for voice), which make the recognition of real objects easier and allow the user to browse among them and choose the desired ones.

The step of presentation of retrieved information must be considered, too. A specific object does not give rise to any problem, as the whole information about its presentation is contained in its layout structure. However, the user might wish to get different instances of objects, perhaps of different classes, and to output them with a certain syncronization. Therefore, inside a request of retrieval, the user should be allowed to specify, in addition to what he wants, also where (e.g. instances of different objects on different parts of the output area), when (e.g. timing among different instances) and which way (e.g. instances of different classes conformed to certain dimensions).

4. Conclusions

In this paper the problems concerning the definition and the management of multimedia information (complex information consisting of conventional data, text, graphics, images and voice) have been analized.

A multilevel approach in conceiving multimedia applications has been proposed where the inner level is represented by a Multimedia Data Base Management System that mainly has the role of conventional DBMS in traditional applications. This way it has been outlined an environment where multimedia information can be handled, stored and retrieved in order to solve problems common to a wide range of application domains and to provide the application programmer with a valid set of primitives.

A multimedia object model based on the same fundamental concepts which Office Automation documents are based on, has been defined. Such concepts have been adapted to the more complex multimedia environment, supporting for example timing and lasting functions, allowing an object to be a component of several ones, and so on.

The major functionalities of a Multimedia Data Description Language and of a Multimedia Data

Manipulation Language for a Multimedia Data Base Management System have also been outlined, even if they have not been described in detail because not fully defined yet.

Acknowledgements

The authors are grateful to their colleague Mauro N. Bert for the numerous and positive discussions during which many obscure points were clarified. He also assisted the effective writing out of this paper.

Bibliography

[ANDR84] Andrews,H.L. "Speech processing", **Computer**, Vol. 17, No. 10, october 1984.

[BARB85] Barbic,F., Rabitti,F. "The type concept in office document retrieval" **Proc. 11th. Int. Conf. on Very Large Data Bases**, Stockholm, 1985.

[BERT85] Bertino,E., Gibbs,S., Rabitti,F., Thanos,C. and Tsichritzis,D. "Architecture of a multimedia document server", **Proc. Esprit Technical Week** 1985, Brussels, september 1985.

[BUCC85] Bucci,P., D'Aguanno,C., Libutti,P. "The multimedia document filing and retrieval system in the SOMIW workstation (in italian)", **Workshop AICA on Text Processing**, Milano, november 1985.

[CELL85] Cellier,A., Chrisment,C., Pujolle,G., Zurfluh,G. "Le systeme GIM", **Proc. CIL 85 Convenzione Informatica Latina**, Barcelona 1985.

[CHRI85] Christodoulakis,S. "Office filing", pagg. 67-90 in Tsichritzis,D. ed., "Office Automation", Springer-Verlag, 1985.

[CRAM83] Crampes,J.B., Chrisment,C.Y. and Zurfluh,G. "The big project", 2nd **International conference on databases**, ICOD2, Cambridge, 1983.

[ECMA85] "Office Document Architecture", **Standard ECMA 101**, September 1985.

[HORA85] Horak,W. "Office Document Architecture and Office Document Interchange Formats: Current Status of International standardization", **Computer**, Vol. 18, No. 10, october 1985.

[LEDI85] Le Dizes,J.M., Madar,J. "A model of computer based multimedia workstation", **Proc. COMPINT 85 Computer Aided Technologies**, Montreal, September 1985.

[LOPE84] Lopez,M. and Velez,F. "Modeling and handling generalized data in the TIGRE project", Working Paper, **IMAG**, St. Martin d'Heres, France, 1984.

[MEYR82] Meyrowitz,N., Van Dam,A. "Interactive editing systems: part I and II", **Computing surveys**, Vol. 14, NO. 3, pagg. 321-415, September 1982.

[NAFF81] Naffah,N., Borron,H., Choquet,N. and Rosset,J.L. "La production des documents dans le projet KAYAK", **Bulletin de liason de la recherche en informatique et automatique**, No. 69, INRIA, 1981.

[POGG85] Poggio,A., Garcia Luna Aceves,J.J., Craighill,E.J., Moran,D. Aguilar,L., Worthington,D. and Hight,J. "CCWS: a computer-based multimedia information system", **Computer**, Vol. 18, No. 10, october 1985.

[RABI85] Rabitti,F. "A model for multimedia documents", pagg. 227-250 in Tsichritzis,D. ed., "Office Automation", Springer-Verlag, 1985.

[SIGM85] "Panel on multimedia database management systems", SIGMOD annual meeting, 1985.

[SMIT84] Smith,J. "Formatting standards for document interchange", **Computer communications**, Vol.7, No. 4, August 1984.

[STAR85] Star,J.L. "Introduction to image processing", **Byte**, Vol. 10, No. 2, pagg. 163-170, February 1985.

[TSIC83] Tsichritzis,D., Christodoulakis,S., Economopoulos,P., Faloutsos,C., Lee,A., Lee,D., Vandenbroek,J. and Woo,C. "A multimedia office filing system", **Proc. 9th. Int. Conf. on Very Large Data Bases**, Florence, 1983.

[TSIC85] Tsichritzis,D. ed., "Office Automation", Springer-Verlag, 1985.

[WOO 85] Woo,C.C., Lochovsky,F.H. and Lee,A. "Document management systems", pagg. 21-40 in Tsichritzis,D. ed., "Office Automation", Springer-Verlag, 1985.

[VLDB83] "Panel on complex data objects: text, voice, images: can DBMS manage them?", **Proc. 9th. Int. Conf. on Very Large Data Bases**, Florence, 1983.

Methodological Issues for the Design of an Office Information Server *

- Focal Topics for the Analysis from an Office System Perspective -

Till W. Truckenmüller

Fraunhofer Institut für Arbeitswirtschaft und Organisation (IAO), Stuttgart
University of Stuttgart
Federal Republic of Germany

Abstract

This paper deal with the necessity of consideration of organizational and user requirements to create the basis for the successfull design of future office information servers.

Today volumes of the order of 10.000 to 50.000 multimode documents and 1 to 10 million documents at a company level (companies with over 1000 employees) per system have to be archived. The average amount of filing is 12 to 16 running metres of paper per year, with an increasing tendency (Ben84).

However the shortcoming of systems of today is not the incapability of storage of big amounts of information but the fact they only support particular (and well structured) office tasks in an operative way. Future systems have to support all kind of work (procedures) people do in an office, consider the strategic and operative goals of the particular office and take the user behaviours especially their knowledge to solve problems and the individual kind of doing their jobs (e.g. search strategies, reminder functions) into account.

Analysis methods for the collection of data for the design of office systems have mostly been developed in the context of office automation to develop systems to support particular tasks, to restructure offices to improve the profitability of the company. Because of that fact, these methods must be quite extended to supply data which are suited for the successfull design and development of future office information servers.

Focal topics of the analysis from an office system perspective are: information generation and location, consistency and permanence, work support, information handling and manipulation, access right and confidentiality, accountability, information flow and use of abstractions. Detailed dimensions concerning these areas are posted in this paper.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

I Successful design of future office systems: The role of organizational and user requirements

Studies done by von Hippel maintain that users are the source of most of the successful innovations (Hipp82, Hipp78a, Hipp78b). One of the most effective ways of creating new products is to acquire an accurate under-standing of user and organizational needs (Hipp76):

* This work was founded by the Commission of the European Communities in the context of the ESPRIT project 231 (DOEOIS), Design and Operational Evaluation of Office Information Servers

- Three out of four commercially successful industrial innovation projects are initiated in response to a perception of user need for an innovation, rather than on the basis of a technological opportunity to achieve them.
- Accurate understanding of the user need is the factor which discriminates most strongly between commercially successful industrial innovation projects and those which fail.

Besides other tasks, this is one of the main tasks of Office System Science of today (Niem86).

Systems which support simple and particular tasks are used in many offices of today. However this level of task support is of limited effectiveness because of the neglect of terms like cost emergence, cost production or value creation.

Successful systems should support the user in a "cognizant" manner and, not as today, in an operative manner (Barbe84). Cognizant systems are systems that have a semantic understanding of the tasks performed and of the operations and capabilities of the organization. As a basis for the development of a task support system, Croft (Croft84) cited the following user requirements: automation of routine tasks, assistance with complex tasks, context switching, handling unusual actions, adaptability, task invocation, status inquiry and explanation and handling multi-user tasks.

Effective systems have to support the office work as an entire unit. Besides the office has to be regarded as a complex and highly interactive system. The empirical office analysis has to be separated into two research areas (Ness86a):

- the first part is the invariant structural part of office work, the basic goals, missions and functions, and their interrelations
- the second part is the one that depends on the special implementation of these functions in an office environment

Most important for the user support is the extension of filing and retrieval capabilities, the most important task to be supported in offices. As a result of the increased use of computer systems the amount of information available has grown. Despite this tendency we can still talk of a "lack of surplus". More and more information is available but the specific aspects of this information required are becoming increasingly detailed. Inadequate or defective information supply can be immensely detrimental to the success of the company as a whole (Niem85).

Thus, the information flow must be reduced to those aspects which are essential to efficient performance of the various procedures. The accessibility of stored or filed data must also be considerably increased. Support in the internal processing of complex tasks through access to common information will therefore be an increasingly indispensable contribution to improving competitiveness.

As far as flexible and rapid reaction to market developments and the processing on time of information is concerned, information and communication systems can be used as strategic instruments for improving the competitiveness (Bull84). Should response capability join price and quality as a factor in competition in the future, the time required for retrieving information and converting it into actions will have to be reduced. The most significant requirements of a technical infrastructure for information supply are shown in the following illustration:

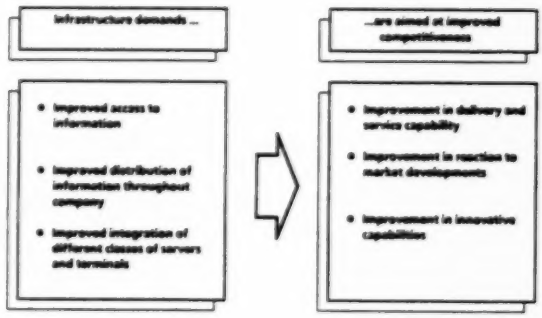

infrastructure demands ...
• Improved access to information
• Improved distribution of information throughout company
• Improved integration of different classes of servers and terminals
...are aimed at improved competitiveness
• Improvement in delivery and service capability
• Improvement in reaction to market developments
• Improvement in innovative capabilities

Figure 1: Technical infrastructure and information supply (see Niem85)

II Filing and Retrieval viewed from an office work perspective

All research done so far in the filing and retrieval area points to a lack of a sufficient amount of qualitative and quantitative data, on which to guide the design of office information servers (Bull86, Truck85). All office organizations are composed of a number of locations where information is generated, processed and stored, which is the basis for the concept of an office information server. An analysis of offices will have to be conducted with the objective of identifying the specific requirements of an office information system. From these requirements, tools and specifications for the design and subsequent evaluation of office systems can be developed.

Existing technological approaches for the design of interfaces of office information systems are (Bull85, Bull86, Croft83):

- the so called "desktop" metaphor systems, a recently introduced user interface; mostly found on personal workstations using a hierarchical filing system. E.g.: XEROX (STAR) (Pur83), APPLE (LISA, MACINTOSH) (Will83, Will84, Triumpf Adler (System M32).

- query systems (for free text searching) using specified character strings connected with Boolean operators. In the office of today this approach is realized in (bibliographic) database systems mainly supporting big amounts of structured data on a company wide or department level (or in a library); mostly hosted by mainframe computers.

Disadvantages of the first approach are: Filing systems are inflexible in nature having no capability for fast and intelligent retrieval of information by content and context of the stored documents. This is especialy a disadvantage when the number of filed documents is getting larger. The effectiveness of these electronic filing systems is particularly poor in non-routine situations, where a heterogeneous group of persons, for example managers, officers, secretaries, clerks, are searching for information at different times, for different reasons (Bull86).

The other approach is also not best suited, if at all, for office applications. Tsichritzis et al. (Tsi85) have reviewed document filing and retrieval methods such as filing by labels, sequential storage, retrieval by keywords or attribute values, retrieval by pattern matching, and encapsulation of document properties in abstractions (see Bull86). Faloutsos and Christodoulakis (Fal85) have compared access methods for formatted data (data attributes) and unformatted data (text). For integrated data and text environments signature file methods performed best in their evaluation, followed by full-text scanning.

Other approaches want to integrate aspects from artificial intelligence and information retrieval to present the characteristics of users in an office (Croft85):

- domain knowledge
- subsequent feedback to construct new models of the current information needed
- search strategies typical for a special user
- browsing

To use organizational and user requirements, is also the basis for the work of Barbic and Illuzi (Barbi85). A good comparison between the main features of Information Retrieval Systems (IRS) and Data Base Management Systems (DBMS) from the office point of view has been posed. The conclusion is to integrate (and extent) DBMS and IRS characteristics, because DBMS as well as IRS provide some of the characteristic features needed in an office message filing system. Features of an office message filing system are: inductive(IRS)/ deductive (DB) inferences, deterministic (DB)/ probabilistic (IRS) model, incomplete request (IRS), insensible error response (IRS), monothetic (DB)/ polythetic (IRS) classification, very high level query language, partial (IRS) and exact (DB) match of retrieval, middle retrieval frequency, high insertion frequency, low deletion frequency (IRS), low update frequency (IRS), recall equal 1 (DB), precision less than 1 (IRS) and typical storage size 10 to 300 Mb (Barbi85).

Looking into offices two characteristic features of office activities can be established:

- lack of structure and
- complexity

In these two aspects are to be found the most significant problems in a description and representation of procedures. The problem of lack of structure is clearly demonstrated when the decision making process is considered and also in the realization that various completely different methods and solutions are possible for the successful completion of an office procedure. The expert knowledge of those involved in the process influences both, the content and the time sequence of the procedure. The complexity of office procedures is a further factor. In many cases an operation cannot be completed within a single department as data from other sectors of the company or from outside required, or else the operation needs to be processed in different places. Added to this is the fact that processes are often extremely interactive and have to be completed within a defined time framework.

Concentrating on the filing and retrieval activities, a number of typical office attributes have to be taken into account. Most offices of today are paper oriented and therefore the basic information carriers are documents (Bull86, Truck85):

- in offices there exists a variety of documents with different content;
- these documents are the result of the work of different persons;
- particular documents are connected with particular office procedures;
- the information in the documents is entered at different times;
- there are varying levels of detail and precision in the information contained in a document.

Derived from these characteristic features of office activities it is easy to understand why procedural approaches often fail. Problems which arise by using procedural descriptions are also well known and described in the literature (e.g. Barbe84, Truck85).

- procedural descriptions are only useful to give general ideas of what goes on. Mostly there are too many details to regard and too many conditions and exceptions that cannot been foreseen
- the description of tasks and what actually happened in a particular procedure are very different
- a description of a procedure gives only a determination of what happend, not why it happend
- exeption handling and evolution cannot be managed
- informal information flow cannot be seized
- relevance of information or procedures cannot be established

Today data and documents are typically organized in a variety of filing systems (Bull86, Truck85):

- Company level:
  centralized archives, reference libraries for specified groups of persons (passive information, e.g. documentations, financial information; active information e.g. address lists, telephone numbers, price lists)
- Department level:
  decentralized archive systems, usually located near the workplace (e.g. special price lists, department dependant instructions)
- Personal Worksystem:
  desktop archive system, which consists normally of a variety of trays, box files, dossiers etc. (e.g. forms, process dossiers)

A study by Malone (Mal83) looks into the way offices and desks of today are organized with respect to filing and reminder functions. The aim of the study was to derive a user profile for the design of future office systems (see Niem85). According to Malone the two essential units for office filing are "file" and "pile". Files are units whose elements (e.g. folders, documents, etc.) are explicitly labelled and systematically arranged. Piles, on the other hand, have non-explicitly labelled units arranged in a particular order. The important feature of piles, however, is their position in

the personal information environment as a memory aid for information and procedures.

Why do office workers arrange their personal work environment in "disorganized" piles? Malone (Mal83) cites four reasons:

1. Explicit classification is difficult to change at a later date, particularly if it extends over several levels. Where no long-term and final filing structure exists there is a great danger that future reorganization of the existing structure will be too time-consuming.

2. When working it is often difficult to devise categories and classifications in such a way that information can be retrieved without difficulty at a later date.

3. Piles often serve as reminder mechanisms for work still in hand. If the piles are removed from the personal work enviorment the work in hand is forgotten.

4. Piles often contain information which is used frequently and which the worker needs continuous access to.

However the organization of information in piles has considerable disadvantages. The more information stored in piles the more difficult it becomes to retrieve information. As the size and number of piles increases the reminder function is drastically reduced because of the arrangement of the information environment (we think that most of the search operations in an office are content oriented).

Documents are often sought in offices by leafing through files or piles page by page. This search mechanism is always used when content-oriented search operations are called for. Industrial research laboratories are currently investigating hardware solutions for such tasks in the form of file filters. A typical example of this is the CAFS (Content Addressable File Store) from ICL and SCHUSS (Naff85) which is used in office work station development as an associative search module. A generally accepted document architecture is, however, required before special processors can be used economically for content-oriented search functions.

In the field of document architecture great efforts are being made at the moment to provide some kind of standardization. Internal standards such as Document Interchange Architecture (DIA) and Document Content Architecture (DCA) of IBM (Völ84) specify document interchange and structural information. The European Computer Manufacturers Association (ECMA) is currently devising a standard to facilitate the exchange of office documents (ECMA85, Krö84, Hor85a, Hor85b, Blum85).

However the degree to which such standards are adopted will depend upon how closely they are able to accommodate actual user and organizational requirements. Since these are still far from being completely identified the uptake of current standards, although fulfilling a necessary requirement, for document interchange can be expected to proceed with understandable caution.

III Detection of functional requirements of an office information server: Demands for analysis methods

In the late 1970s and early 1980s a number of models for office describing activities in the context of office automation have been developed. These models have since led to the design of corresponding office analysis methodologies.

The lack of structure as most typical feature of office activities is the greatest problem in the representation of analysis instruments (see Bull86, Truck85). The first developments in office analysis methods therefore used processes which could be structured, particularly as such methods could often be situated within the sphere of conventional computer systems descriptions. Recently, some more comprehensive methods have been proposed to carry out an analysis of office work resulting in a structured description of the

"office" in terms of the underlying model (see Brac84, Krall84, Palm84, Truck85). The descriptions obtained can then be used as a methodological basis for the development of design criteria of an office information server. However, no generally acceptable office systems design methodologies have been developed to date. A state-of-the-art review and evaluation of office analysis methodologies, office description models and office automation concepts is given in (Truck85).

Extending the work of Pick and Whinston (Pick84), five major classes of evaluation criteria have been identified as direct relations between the objects: office, theory, analysis method, analysis results, and usage of the results.

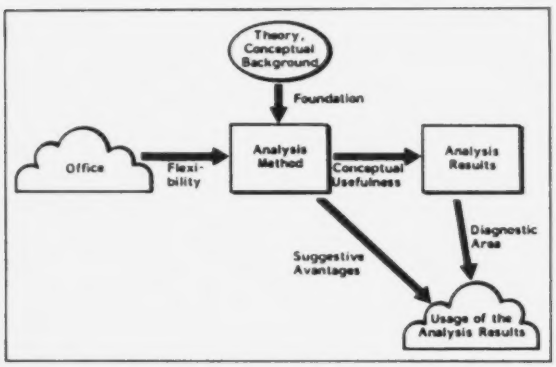


Figure 2: Framework of the five classes of criteria for the evaluation of office analysis methods (Bull86, Truck85)

The Foundations can comprise both a mathematical description or theory and a conceptual background of the method (see Bull86, Truck85). Flexibility is measured with respect to the office activities that can be analysed. Conceptual usefulness asks for "office work-related constructs", "hierarchical breakdown of office work", "top-down" or "bottom-up", identification of relative importance of the different objects in an office, "latitude for interpretation", and "low sensitivity to analyst". By the phrase "Suggestive Advantages of the Method" is meant that the application of the method itself, not the use of its results, generates a greater understanding of the office, suggests different points of view, and contains elements that allow a comparison of organizations. None of the methods and models investigated here enable a comprehensive and flexible analysis of all aspects of office work, rather they have been designed to address specific sections of office work more effectively.

IV A practicable four-level methodology

Two possible methods of approach may be taken into account to answer the questions formulated above:

- The development of a new method to register the field data necessary for specifying future office information servers
- The construction of a method concept from existing methods through considerable extension and adaption

The second method appeared more practical to us for the following reason:

- Experience gained to date can be used in the further development of existing methods. The period from the development of a method concept to its implementation in the field is thus considerably shorter than if a new method is devised and tested from scratch.

A four-level methodology for office analysis consisting of two qualitative and two quantitative approaches has been developed integrating well established methods. Critical Success Factors (Bulle81, Boyn84, Rock79, Rock82, Rock84) and Office Analysis Methodology (Kuni82, Sirbu83, Suth83) will be applied to identify in a qualitative fashion the vital information needs on the company level and the functional office structure on the work system level, respectively. On the information flow level Augmented Petri Nets (Rei85, Rich83, Rich84) will be used to model the information flow in order to obtain quantiative data about filing and retrieval transactions and information containers.

For the analysis of office documents a set of criteria, based on the proposed ECMA and ODA standards has been developed. The set has been extended to accomodate input derived from information flow and criteria related to organizational factors. This information will then be used for the statistical analysis of a large population of office documents collected in a variety of field sites in order to identify typical information entities and their frequency of occurance. In particular, typical office documents can be recognized, and, based on these, requirements of an office information server can be derived. The flow chart of the developed four-level methodology is shown in figure 3.

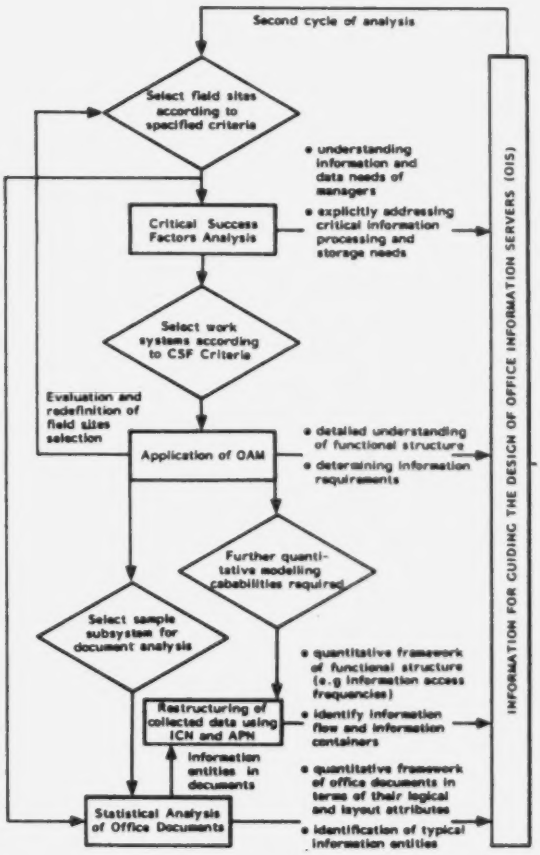

Second cycle of analysis
Select field sites according to specified criteria
• understanding information and data needs of managers
Critical Success Factors Analysis
• explicitly addressing critical information processing and storage needs
Select work systems according to CSF Criteria
Evaluation and redefinition of field sites selection
Application of OAM
• detailed understanding of functional structure
• determining information requirements
Further quantitative modelling cababilities required
Select sample subsystem for document analysis
• quantitative framework of functional structure (e.g information access frequencies)
Restructuring of collected data using ICN and APN
• identify information flow and information containers
information entities in documents
• quantitative framework of office documents in terms of their logical and layout attributes
Statistical Analysis of Office Documents
• identification of typical information entities
INFORMATION FOR CUIDING THE DESIGN OF OFFICE INFORMATION SERVERS (OIS)

Figure 3: Procedure of the four-level office analysis methodology (Bull86, Truck85)

Nevertheless, the work we have done in a lot of field sites and the deduced above described organizational and user requirements necessitate the extension of the methods in some major points to provide answers to the information needs of system developers. These information needs will be described in the next section.

V Focal points to be considered as a necessary information for the design of office information servers

Which data do we want to collect?

- data which are necessary for system designers to develop a system which support all requirements users have to an electronical office system. The system has to support all the functions (also reminding functions) a user normally uses and needs for doing his work. All organizational behaviours have to be regarded.

The necessary information for the design of office information servers and which of the (extended) methods can be used for the data collection in field sites is shown in figure 4.

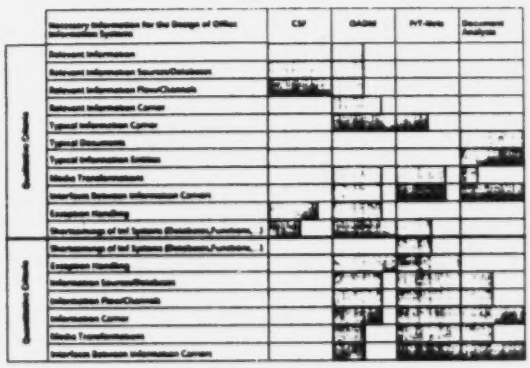


Figure 4: Necessary information for the design of office information servers.

We know that actually we need for the design of future systems the real world data of organizations of the future. But of course we are not able to look in the future, therefore we have no better way as to extract

the information wanted from offices of today and the data which can expected to be true in the future. This leads to the necessity of using more future oriented methods like szenario setting etc. in the next steps of the project. Nevertheless, the analysis of data of today offices leads to a solid foundation of the design work.

The organizational and user requirements analysis have been feasibled along a set of questions respectively research areas (see Ness86a, Truck86a). In other words: The par-ticular topics reflect the requirements from the user and organization point of view. Regarding the answers for the development of future office information servers, a system which can support office workers in an optimal way, will be the result.

Focal topics of research for the design of future office information servers
Information handling and manipulation
Access rights and confidentiality
Consistency and permanence
Information generation and location
Information flow
Work suport
Accountability
Use of abstractions (classification, generalization, association)

Figure 5: Focal topics of research for the design of future office information servers

All these topics can be discussed in a variety of aspects. As an example, the aspects of three topics have been listed (Ness86a,Ness86b, Ness86c, Truck86a,Truck86b):

- information handling and manipulation
  - which functions does an office information server have to support (storage, retrieval, recording the state of a procedure, controll processes, invoking of prodcedures/activities, deletion, ...)
  - content / logical / layout characteristics of information entities used in queries (most search is content/process oriented)
  - which information is current (close to the worker)
  - which information is convenient
  - characteristics of dead information
  - handling of states and history (e.g. roll back necessity)
  - handling of time (relative, absolute (time triggers, periodical time triggers), time ranges/ periods, parallel processing, time-shifts)
  - life time of information entities (ranges) (dependant from storage location and content)
  - information handling over the life time of information entities (add, insert, compress, copy, create, interprete, merging with other information entities, modify, ...)
  - levels of importance of information entities (related to access frequency? (e.g. memos are devided in: for information, for reminding, to invoke an action)

- access rights (authorization) and confidentiality
  - different types of access to information (by name, date, content, layout, ...)
  - do access rights depend on the role, position, hierarchy level, type of organization (functional, matrix, project, divisional, ...)
  - relation between reporting structure and information access
  - how far up and down should one have access to information
  - information access to peers on the same level
  - stricly private (confidential) information

- consistency and permanence
  - importance of consistency (in what kind, for what time, in which type of procedure, ...)
  - importance of permanence (kind of information entity, kind of procedure, ...)
  - mechanisms to provide consistency and permanence (today)
  - frequency of versions (creation, manipulation, deletion, dependant from e.g. the hierarchy level, ...)
  - handling of older versions

- information access to an older data base state (in the case of system failure, back tracking, legal necessities, ...)
- level of consistency during the execution of office procedures (e.g. long lived transactions, locking constraints)
- locking necessities (e.g. classes of procedures, cer-tain parts in a procedure)
- failure transpareny (recovery process after failure)
- monotonic version management

This paper represents the ongoing work of the ESPRIT project 231, Design and Operational Evaluation of Office Information Servers. The next step will be the formulation of design guidelines for the development of office information servers as basic information for system designers.

Acknowledgements

I would like to thank Mr. Joachim Niemeier, Mr. Andreas Ness and Mr. Michael Rathgeb (IAO Stuttgart, University of Stuttgart) for their comments and helpful suggestions.

References

Barbe84: Barber, G.R.: An Office Study: Its Implementations on the Understanding of Organizations, in: Ellis, C.A. (Ed.), Second ACM-SIGOA Conference on Office Information Systems, Toronto, 1984

Barbi85: Barbic, F.; Illuzi, S.: An Office Message System, in: RIAO85, Grenoble, France, 1985

Ben84: Benke, H.: Einführungsstrategien für integrierte Bürosysteme, in: Bullinger, H.-J. (ed.), Integrierte Bürosysteme, Springer Verlag, Berlin, Heidelberg, New York, Tokyo, 1984

Blum85: Blumenfeld, M.: Dokumentenarchitektur und Austauschformate, in: Wedekind, Kratzer (eds.), ACM Tagung Büroautomation 85, Erlangen, 1985

Boyn84: Boynton, A.C.; Zumel, R.W.: An Assessment of CSF; in: Sloan Management Review 1984, pp. 17-27

Brac84: Bracchi, G.; Pernici, B.: The design requirements of office systems in: ACM Transactions on Office Information Systems, Vol. 2, No. 2, April 1984, pp. 151-170

Bulle81: Bullen, C.V.; Rockart, J.F.: A Primer on Critical Success Factors. Center for Information Systems Research, Sloan School of Management, Massachusetts Institut of Technology, 1981

Bull84: Bullinger, H.-J.: Die Durchdringung der Unternehmen mit integrierten Bürosystemen, in: Bullinger, H.-J. (ed.), Integrierte Bürosysteme, Springer Verlag, Berlin, Heidelberg, New York, Tokyo, 1984

Bull85: Bullinger, H.-J.; Fähnrich, K.-P.; Ziegler, J.: Xerox 8000 Netzwerksystem im praktischen Einsatz. Fraunhofer Institut für Arbeitswirtschaft und Organisation, Rank Xerox GmbH (Ed. 1984)

Bull86: Bullinger, H.-J.; Fähnrich, K.-P.; Niemeier, J.; Ness, A.J.: From research to product design: The office information file server example, in: ESPRIT85 status report of continuing work. The Commission of the European Communities (Eds.), Elsevier, 1986

Croft83: Croft, W.B.: Applications for Information Retrieval Techniques in the Office, in: Sixth Annual Intern. ACM SIGIR Conference, Maryland, 1983

Croft84: Croft, W.B.; Lefkowitz, L.S.: Task Support in an Office System, in: Ellis, C.A. (ed.), 2. ACM-SIGOA Conf. on Office Information Systems, Toronto, 1984

Croft85: Croft, W.B.: An Expert Assistant for a Document Retrieval System, in: RIAO85, Grenoble, France, 1985

ECMA 1985: ECMA: Office Document Architecture, Final Draft, TC 29; April 1985

Fal85: Faloutsos, C., Christodoulakis, S.: Access Methods for Documents, in: Tsichritzis, D. (ed., 1985), pp. 317-338

Hipp76: Hippel, E. von : The Dominant Role of Users in the Scientific Instrument Innovation Process, in: Research Policy, 5, 1976, pp. 212-239

Hipp78a: Hippel, E. von: Useres as Innovators, in: Technology Review, January 1978, pp. 31-39

Hipp78b: Hippel, E. von: Successful Industrial Products from Customer Ideas, in: Jour. of Marketing, pp. 39-49

Hipp82: Hippel, E. von : Get New Products from Customers, in: Harvard Business Review, March-April 1982, pp. 117-122

Hor85a: Horak, W.: Office Document Architecture and Office Document Interchange Formats: Current Status of Int. Standardisation, in: Computer, Oct. 1985, pp. 50-60

Hor85b: Horak, W., F. Tartanson, and G. Coulouris : Handling of Mixed Text/Image/Voice Documents Based on a Standardized Ofice Document Architecture, in: Roukens, J., J.F. Renuart (eds., 1985), pp. 395-410

Krall84: Krallmann, H.:Lokale Netze und öffentliche Dienste, in: Bullinger, H.-J. (ed.), (Bull84)

Krö84: Krönert, G.: Textverarbeitung auf Basis der standardisierten Dokumentenarchitektur - Eine Einführung, in: Schindler, S. (ed.), Berlin, 1984

Kuni82: Kunin, J.S. : Analysis and Specification of Office Procedures, Massachusetts Insitute of Technology, Laboratory for Computer Science, MIT/LCS/TR-275, Cambridge, Massachusetts 1982

Mal83: Malone, T.W.: How Do People Organize Their Desks? Implications for the Design of Office Information Systems; in: ACM Transactions on Office Information Systems, Vol 1, 1983, No. 1, pp. 99-112

Naff85: Naffah, N.; Kempen, G.; Rohmer, J.; Steels, L.; Tsichritzis, D.; White, G.: Intelligent Workstation in the Office, State of the Art and Future Perspectives, in: Roukens und Renuart, (eds.), 1985, pp. 365-378

Ness86a: Ness, A.J.; Rathgeb, M.R.: Questions for the Organizational and User Requirements Analysis, Working Paper IAO-ANN-863-2, 1986

Ness86b: Ness, A.J.: Attributes of the Office, Working Paper IAO-ANN-865-1, 1986

Ness86c: Ness, A.J.; Niemeier, J.; Truckenmüller, T.W.: Elektronische Ablagesysteme: Von der Bürosystemforschung zur Produktgestaltung, Paper accepted at: ÖGI/GI/SI Conference, University of Linz, Austria, Sept. 1986

Niem85: Niemeier, J.: ... und die Datenverwaltung? Aktenordner oder elektronische Ablagesysteme, in: IRD - Fachtagung Bürokommunikation, Stuttgart, September 1985

Niem86: Niemeier, J.: Bürosystemforschung - Auf dem Weg zu methodengestützten Analyse- und Gestaltungskonzepten, in: 3. Int. Kongreßmesse für Technische Automation, Essen, Germany, 1986, pp. 1K1-1K26

Palm85: Palm, S. U. : An Office Investigation and Analysis Method, in: Roukens, J.; Renuart, J.F. (eds., 1985), pp. 283 - 294

Pick84: Pick, R.A.; Whinston, A.B.: A Language for Modeling Organizational Activities, Graduate School of Business, University of Wisconsin, Madison, Wisconsin Working Paper 11-84-24, November 1984

Pur83: Purvy, R.; Farrel, J.; Klose, P.: The Design of Star's Records Processing:Data Processing for the Nonprofessional, in: ACM Transactions on Office Information Systems, 1, 1983, pp. 3-24

Rei85: Reisig, W.: Systementwurf mit Netzen; Springer Verlag, Heidelberg, New York, Tokyo, 1985

Rich83: Richter, G. : Realitätsgetreues Modellieren und modellgetreues Realisieren von Bürogeschehen, in:Wißkirchen, P. (Ed., 1983),pp. 145-214

Rich84: Richter, G. : Netzmodelle für die Bürokommunuikation, Teil 2, in: Informatik Spektrum 7, 1984, pp. 28-40

Rock79: Rockart, J.F. : Chief Executives Define Their Own Data Needs, in : Harward Business Review, March-April 1979, pp.81-93

Rock82: Rockart, J.F.: The Changing Role of the Information Systems Executive: A Critical Success Factors Perspective; in: Sloan Management Review, Fall 1982, pp. 3-13

Rock84: Rockart, J.F. and A.D. Crescendi : Engaging Top Management in Information Technology, in: Sloan Management Review, Summer 1984, pp. 3-16

Şirbu83: Sirbu, M.A. et al.: Office Analysis: Methodology and Case Studies, Massachusetts Institut of Technology, Lab. for Computer Science, MIT/LCS/ TR-289, Cambridge, Massachusetts, 1983

Suth83: Sutherland, J.: An Office Analysis and Diagnosis Methodology, Massachusetts Institut of Technology, Lab. for Computer Science, MIT/LCS/TR-290, Cambridge, Massachusetts, 1983

Truck85: Truckenmüller, T.W.; Ness, A.J.; Niemeier, J.: An Analysis Method for Office Automation, Deliverable ESPRIT-Project 231 (DOEOIS), 1985

Truck86a: Truckenmüller,T.W.; Ness, A.J.; Rathgeb, M.R.: Questions for the Organizational and User Requirements Analysis (Revised and extended version), Working Paper IAO-TT-865-1

Truck86b: Truckenmüller, T.W.; Rathgeb, M.R.: Time Management, Version Management, Access Rights, Working Paper IAO-TT-866-1

Tsi85: Tsichritzis, D.; Christodoulakis, S.; Vandenbroek, J.: A Multimedia Filing System, in: Tsichritzis, D. (Ed.), Office Automation, Concepts and Tools, Berlin, Heidelberg, New York, Tokyo, 1985

Völ84: Völcker, U.: IBM - Architekturen für Bürokommunikations-Systeme, in: Krallmann, H. (Hrsg.), 1984, S. 57-69

Will83: Williams, G.: The Lisa Computer System, in: BYTE, 8(1983)2, pp. 33-50

Will84: Williams, G.: The Apple Macintosh Computer, in: BYTE, 9(1984)2, pp. 30-54

IR, NLP, AI and UFOs: or IR-RELEVANCE, NATURAL LANGUAGE PROBLEMS, ARTFUL INTELLIGENCE AND USER-FRIENDLY ONLINE SYSTEMS

Tamas E. Doszkocs

National Library of Medicine, USA

Abstract

User Friendly Online Searching is examined in the context of Natural Language Processing in Information Retrieval and Artificial Intelligence. Opportunities for synergetic R & D are identified as the basis for Intelligent Information Retrieval and Artificial Retrieval Intelligence.

INTRODUCTION

Information Retrieval (IR) systems are the only practical domain independent systems available currently for the efficient searching of large bodies of unstructured natural language text.

IR systems achieve remarkable search speed and flexibility despite the virtual absence of formal language analysis procedures and meaning interpretation of the underlying text content (1,2). State-of-the-art IR systems pragmaticly blend the best features of diverse probabilistic-combinatoric and Boolean logic retrieval models and readily support free-form natural language user interfaces(3,4,5). Yet, such direct natural language interfaces need to incorporate more sophisticated natural language processing (NLP) and other Applied Artificial Intelligence (AI) techniques in order to intelligently cope with the inherent ambiguities of natural language queries and text, and to compensate for the inevitable semantic loss and confounding inherent in indexing and query matching (6).

By the same token, AI approaches to natural language processing, particularly as applied to NL user interfaces and text searching, have much to learn and benefit from proven IR concepts and techniques in order to transcend still prevalent domain-specificity and performance problems (7). Thus natural language databases and user friendly online searching (UFOS) represent challenging and mutually supportive common problem areas for both IR nd AI.

INFORMATION RETRIEVAL Although a variety of access methods have been developed for text retrieval (8), operational IR systems are, almost without exception, characterized by the dominance of the inverted file, Boolean logic search paradigm (1). In their basic form, IR systems are designed to manipulate fundamentally simple natural language (NL) text structures, such as bibliographic citations or full text documents, although some IR software packages have been enhanced to incorporate generalized DBMS access methods and special processing functions needed for the handling of integrated textual, numeric, graphics and image data. The automatic "indexing rules" or algorithms are, by and large, lexically based procedures guided by delimiters and/or lists of non-indexed "stopwords", and the resulting inverted search keys are typically organized into B-tree structures for acceptable trade-off between speed of access and ease of updating. This approach is, for practical purposes, highly flexible and domain independent, although distinct indexing rules are needed in order to usefully fragment special textual fields, for instance chemical names. In addition, in most IR systems, the automatically generated keywords are frequently augmented with human assigned subject headings or thesaurus entries, mostly noun phrases, for added syntactic and semantic precision in searching.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

Search queries, regardless of their form of input, must be ultimately transformed (typically by trained intermediary searchers) to conform to the basic indexing scheme of the given IR system. Query search keys are matched against the inverted file search keys, and the corresponding ordered inverted lists are are compared using exact match Boolean AND, OR, AND NOT logic operators implemented as set operations (intersection, union and complement, respectively) on the inverted lists. The process is exceedingly fast and efficient as implemented via currently available hardware and software architectures.

The lack of linguistic and cognitive analysis procedures at indexing time, and the resulting precision/recall problems are, to some extent, alleviated by the availability of powerful pattern matching functions and metric (hierarchichal/positional) operators, such as character masking, truncation and adjacency.

Trained intermediary searchers, in turn, provide the needed augmented "knowledge base", "inference engine" and "control strategy" for the proper configuration and sequencing of retrieval operations.

In the last few years, operational IR systems have been implemented that reflect the influence of G. Salton's seminal work and incorporate many useful ideas and results from theoretical and experimental IR research going back to the mid-1960's (3,4,5).

IR research on the whole has been dominated by non-linguistic, primarily lexical-statistical NLP methods as exemplified by Salton's SMART system (9) and the work of researchers such as Bookstein, Kraft, Rijsbergen (10,11,12) and many others. Although experimental IR research prototypes have often suffered from problem of scale limitations in their technical approaches, IR research has nontheless contributed a coherent conceptual framework and identified a core of desirable IR functions and evaluation methods that are of particular relevance to the design and implementation of end user oriented information retrieval systems and services. The most important among these are the notions of unrestricted natural language query input, closest match search strategy, the ranking of the retrieval output according to expected relevance to the query and the dynamic utilization of user feedback in automatic query reformulation and search strategy modification.

Large scale implementations of NL IR interfaces, such as CITE (13), efficiently combine the best features of the Boolean and the probabilistic/combinatoric retrieval models with limited "intelligent" computational linguistic analysis and AI-type search heuristics.

Such end user oriented systems treat unrestricted natural language both in queries and text records as the least common denominator among different searchers, databases and IR systems, offering the potential of true transportability and transparency among diverse users, information sources and search systems. In order to successfully emulate the trained searcher's augmented retrieval "knowledge base", "inference engine" and "control strategy", however, UFOS must possess intelligent NLP capabilities of greater sophistication (6).

The trend toward full text databases, Very Large Data Bases, mixed information sources containing text, numerics, graphics, image and other data, the increased diversity, depth and breadth of information sources, hybrid technologies (e.g. CD_ROMs), special purpose IR hardware, non-von-Neumann computer architectures,associative memories, distributed and parallel processing and, last but not least, interactive and user searching and personal, computer enhanced "information metabolism" clearly demand the augmentation of conventional information retrieval and filtering capabilities with appropriate and efficiently implemented techniques adapted from AI application areas, such as natural language processing and understanding, expert systems, intelligent information management, and intelligent problem solving.

ARTIFICIAL INTELLIGENCE

Artificial Intelligence, particularly Applied Artificial Intelligence, is a growth industry of high expectations (7). Notwithstanding the difficulty of defining intelligence, let alone artificial intelligence, and despite the many lingering doubts and the seeming Tower of Babel situation, the AI field possesses a healthy basic research underpinning and the AI community has been successful in developing important concepts, tools, techniques and applications of interest to other information intensive disciplines (14,15,16,17,18). Just as "conventional" programming languages and diverse data representation models in "classical" IR and DBMS systems serve as problem solving tools for a wide variety of applications, AI knowledge representation models, such as production rules (19), frames (20) and semantic networks as well as AI programming languages , such as LISP (list processing), PROLOG (logic programming, see (21), OPS5 (22) (rule based programming), SMALLTALK (object oriented programming) and other general purpose techniques and tools (e.g. heuristic search, ATN parsers and rapid prototyping (7.1)) represent versatile AI implements for a broad range of applications, including IR.

Natural Language Processing/Problems

NLP and computational approaches to dealing with natural language were among the earliest objects of interest in AI research. The well known ambitions and subsequent failures of machine translation R & D projects in the 1960's, as well as the recent successes of more limited yet highly pragmatic computer assisted language translation systems epitomize both the great difficulties involved in coping with NL and the degree of maturation and realistic goal setting of the field.

Language understanding, though it appears to humans to be naturally easy, is a difficult task that involves highly cognitive and not yet totally understood intellectual and psychological processes (7.3). Schank (7.3) argues that NLP is both a linguistic and a cognitive task, and it cannot occur without a knowledge base concerning the relevant subject area. This in turn points to a synthesis of natural language processing and expert systems techniques.

Typical areas of concern and investigation in AI NLP research involve automated lexical, syntactic and semantic analysis; dealing with ill-formed or fragmentary input; ellipsis; conjunction and negation; pronoun/anaphora resolution; definite noun phrases, quantification; beliefs and intensions; fail-soft recovery; space and time and contextual understanding - to name just a few.

[The role of NLP in IR is less clear (3), given that the very nature of the textual documents traditionally dealt with in IR does not warrant elaborate analysis, and the fundamentally mechanistic techniques developed in IR have served to handle the passive query-to-document matching problem at an acceptable level of success. The advent of unmediated, "user-friendly" search interfaces on the one hand, and the considerably broader scope and depth of the today's full text data bases, on the other, however, necessitate a careful reexamination of this role].

Computational linguistic processing has reached fairly reliable stability and practical utility in morphological and syntactic analysis.

While procedures for morphological analysis are decidedly non-trivial, they are generally more straightforward and efficient than syntactic analysis. Common strategies for morphological analysis employ some or all of the following: morphological rewriting rules, dictionary lookup, inflection generator, complexity testing and idiom and compound recognition (23,1). Prefixing and suffixing are the main concern in processing English. Often, in processing specialty languages, e.g. medical English, special attention must be paid to characteristic prefixing, suffixing and morphosemantic problems (24). Compare, for instance:

ALEXIA - DYSLEXIA
VITAMINS - AVITAMINOSIS
ANXIETY - ANTIANXIETY AGENTS
INFECT - DISINFECT
HYPERTENSION - HYPOTENSION
ADJUSTMENT - MALADJUSTMENT
ANTIBIOTICS - "-MYCINS"
INFLAMMATION - "-ITIS"

Reasonably , i.e. relatively robust and efficient NL syntactic parsers have been developed and incorporated into many AI applications. "Parsing efficiency is crucial when building practical natural language systems. This is especially the case for interactive systems, such as natural language database access, interfaces to expert systems and interactive machine translation" (25, p. xvii). Tomita's LR context-free parsing algorithm, for instance, takes advantage of the lef-to-rightness of natural language user input, and parsing starts as soon as the user types the first word, thus reducing apparent response time from the user's point of view.

Despite significant advances, a number of unsolved problems and limitations in AI NLP remain. Most importantly, none of the systems developed to date are fluent in the use of unrestricted natural language (7.2). For potential IR use of AI NLP techniques, it is important to remember that while most full sentences are unambiguous, their component parts are frequently ambiguous. Since IR systems utilize component or fragmentary lexical, syntactic and semantic units as a result of indexing, and since indexing inevitably implies omission, the disambiguation problem in domain independent IR systems is much less tractable than in narrow domain AI applications.

Accoustic-phonetic, lexical, syntactic and pragmatic language complexity, as reflected in word sense ambiguity, structural ambiguity, and the referential ambiguity of noun phrases, pose very challenging problems in user friendly AI and IR interfaces. Few if any existing NLP systems, for instance are able to recognize and disambiguate compounds (e.f. database vs data base), acronyms (e.f. S-DUIT), or abbreviations (e.f. MD ==> Maryland ==> physician) in large textual databases.

Noun phrase ambiguity may be further compounded in keyword indexing by ignoring word order, field and other text boundaries, and by partial match search strategy in query matching, e.g.:

MEDICAL LITERATURE ==> MEDICINE IN LITERATURE ==>
LITERATURE IN MEDICINE

SEXUAL PERVERSION ==> SEXUAL ABSTINENCE

There is a lot of humor in all of this, as in:

HUMOR IN MEDICINE ==> AQUAROUS HUMOR IN MEDICINE,

or put differently, language understanding can be a joke (26,27). For proper perspective it is worth keeping in mind that there still does not exist any NLP computer program that can handle nearly all of English syntax, none can even come close to coping with the semantics of all of the English language and none are on the horizon (28). APPENDIX A shows a not unusual example of "wretched and confusing prose" that defeats even human comprehension.

The impossibly large number of rules that would be necessary for the morphological, syntactic and semantic disambiguation of natural language in real-life multi-disciplinary knowledge domains precludes the practicality of the rule based expert system approach as well. [No accurate estimation has been made on how many context free rules are needed to cover English almost completely, but the number is very large (29)].

From the survey of the literature it appears that AI is not quite ready to field any system flexible enough for mass use, save a few relatively small problem domains (7.4).

Can IR facilitate finding AI solutions to practical NLP problems? Certainly so! Domain independent IR search techniques can serve as efficient filters for more refined in-depth natural language processing. As more and more powerful AI concepts and tools become available (e.f. AI PC toolkits) to more and more end users, the problem of scale and performance limitations that characterize contemporary AI systems will be gradually overcome. To the extent that the AI and IR communities will be able to learn from each other and gain mutual insights into the problems of language and searching, there will exist intelligent IR systems and effective domain independent AI NL search systems. And perhaps, in theory at least, the distinctions between the two (mind)sets will be "fuzzy" at best (29) .

NATURAL LANGUAGE and USER FRIENDLY ONLINE SEARCHING (UFOS)

Natural language interface technology represents a major breakthrough in "user friendly" computer systems (7.3). Along with other end user oriented interface techniques, such as menus, windowing, graphics, icons, pointing, touching, etc., commercial implementations of NL interfaces, such as Artificial Intelligence Corporation's INTELLECT or Texas Instrument's Human DBMS front end products target the largest hitherto untapped segment of the information market place, namely end users. The same holds true for public access online catalogs in libraries and for user friendly IR interfaces in general (27), and for NL IR interfaces in particular, like the National Library of Medicine's CITE system for searching the world's largest medical literature databases, MEDLINE and CATLINE (30).

Online systems and the personal computer revolution have made computer resources universally available. Now a similar revolution in software (i.e. user interfaces) is needed to make the computer universally usable (7.5). NL interfaces in IR and AI are the scouts, shock troops, vanguards and sometimes martyrs of the user interface revolution.

With occassional end users already outnumbering trained professional searchers in the user populations of online information utilities like the SOURCE and COMPUSERVE, it becomes increasingly important to develop and refine analytical cognitive models to better assess the user's skills and understanding of the information stored, and to match the user's cognitive model to the system's model and knowledge representation.

Naturally, natural language is not always natural for a user interface (31), but it is particularly well suited for IR and DBMS interfaces due to the very large number of potential users, high volume of query transactions, distribution of costs over large numbers of users, and the fundamentally linguistic nature of the user-system information exchange (33.3, 7.3).

Ease of learning and use and transportability are among the most attractive features of NL front-ends. Natural language shifts the burden of understanding from the user to the system thus allowing the user to focus on the problem at hand (7.5). At the current state-of-the-art, the high development cost and less than 100% reliability of knowledge-based domain specific NL DBMS systems (e.g. EXPLORER, developed by Cognitive Systems, Inc. for oil exploration), appear to have prevented their widespread development and commercial use. Considerably more commercial success has been achieved by domain independent NL DBMS front-ends. INTELLECT, for example, substitutes knowledge of the physical and logical structure of the database and the topology of user interactions and system functions for domain specific semantic knowledge, using the inverted DBMS index as lexical pointers to the system's rather small domain specific and semantically augmented dictionary. Despite its shortcomings in ambiguity resolution (7.3), INTELLECT succeeds in its practical utility, reliability and operational performance.

It is interesting to note here that while a DBMS NL front-end like INTELLECT has to cope with the full spectrum of linguistic processing problems of a "habitable subset" of the English language, namely the limited domain of the DBMS command language and a relatively small number of query interaction paradigms, it does not have to deal with language ambiguity and matching problems at the level of the database content, due to the fact that DBMS systems deal, for the most part, with discrete and finite non-textual data.

By contrast, a NL IR textual DB interface, like NLM's CITE system (13) must focus on language ambiguity and matching problems for all of English. At the same time, CITE need not invest a great deal of effort in full scale linguistic analysis in query-to-command language translation due to the relatively small number of IR commands available, and the simplicity of the underlying database structure. [Although it would be perfectly feasible and appropriate to use INTELLECT-like NLP techniques, instead of some choices, in CITE to "understand" the type of query at hand, identify its topical component, as well as any implied or explicitly stated limitations as to type of material desired, language restrictions, currency of material etc.].

Natural Language Queries in NL Databases

NL IR interfaces must then deal with the problem of language ambiguity at the level of both the query and database content, and must resolve the matching problem between free-form queries and the database in a manner acceptable to end users who typically approach a natural language system with high expectations of (artificial machine) intelligence.

The basic problem areas of matching can be conveniently divided into lexical, syntactic, semantic and special concerns.

Lexical Problems

The inverted keyword index of a NL database of one million records is likely to contain in excess of a quarter million distinct lexical words (33, 33). The frequency distribution of these lexical entries will typically conform to the characteristics of the Zipf distribution. This empirical fact has the following practical implications:

1. There will be a few dozen to a few hundred highly posted (i.e. high DB frequency) index entries, many of which will be natural candidates for exclusion from the index (c.f. "stopwords").

2. Approximately half of all the index entries will have a DB frequency of one, and as many as half or more of these may be misspellings. This suggests the incorporation of efficient spelling error detection/correction algorithms and dictionaries in the NL user interface, and in indexing, with the dictionary preferably derived from and dynamically updated in conjunction with the database itself.

3. Knowledge of the frequency distribution of the lexical entries suggests implicit heuristics for automatic search strategy formulation.

The NL interface must be capable of recognizing and properly dealing with:

frequent acronyms, abbreviations, numerals, chemical names, names of people/tradenames etc. , e.g.:

X RAY ==> X-RAY ; CAT SCAN; US ==> USA ==> U.S. ==> U.S.A; AI ==> ARTIFICIAL INTELLIGENCE; VITAMIN B 1 ==> VITAMIN B1 ==> THIAMIN; TYPE 1 ==> TYPE I; FACTOR V; 2,4,5-T ==> TRICHLOROPHENOXYACETIC ACID ==> AGENT ORANGE; EPSTEIN-BARR VIRUS; BARR BODIES; BARRE-LIEOU SYNDROME; GUILLAN-BARRE SYNDROME;

orthographic and phonetic transcribing and transliteration:

TUMOUR ==> TUMOR; GYNAECOLOGY ==> GYNECOLOGIA ==> GYNECOLOGY

idioms and cliches:

OFF COLOR; CHANGE OF HEART; OUT OF SIGHT

slang, lingo, jargon and lore:

POT; SPEED; ANGEL DUST; CRACK

stopwords that are noun phrase components:

VITAMIN A; HEPATITIS A; "TO BE OR NOT TO BE"; THE "ME" GENERATION

compound words:

DATABASE ==> DATA BASE; ONLINE ==> ON-LINE ==> ON LINE; BACKACHE ==> BACK ACHE ==> BACK PAIN

MORPHOLOGICAL ANALYSIS: Morphological analysis (stemming or "conflation") warrants special attention in large scale NL IR interfaces. The automatic identification of lexical roots in inverted file based operational IR systems is the first step in the process of matching query words and inverted index entries. The latter may have been derived from auxiliary vocabularies that serve as semantic DB navigational tools, or from the DB records themselves. The following examples from the MEDLINE and MEDICAL SUBJECT HEADINGS inverted indices illustrate over present and familiar lexical ambiguities and semantic noise introduced as a result of using word roots with variable length masking operations in matching:

ACCESS ==> ACCESSORY

ASPIRIN ==> ASPIRATION

AUDIT ==> AUDITORY

BATTERED ==> BATTERY

COMMUNICABLE ==> COMMUNICATIONS

CREATINE ==> CREATIVENESS

DIGITAL ==> DIGITALIS

EXTREME ==> EXTREMITIES

EXPECTATION ==> EXPECTORANT

INFANT ==> INFANTILISM

INFORM ==> INFORMAL ==> INFORMER

LABOR ==> LABORATORY

MEDIA ==> MEDIAN ==> MEDIATED

METHOD ==> METHODIST

NIGHTIME ==> NIGHANT

NURSERY ==> NURSES ==> NURSING

RECEPTION ==> RECEPTORS

SHORT ==> SHORTAGE ==> SHORTHAND

TREAT ==> TREATMENT ==> TREATISE ==> TREATY

Short words require even more caution:

AID ==> AIDS

ANAL ==> ANALYSIS

APE ==> APES ==> APEX

ARM ==> ARMY

CARE ==> CAREER

FAIR ==> FAIRS

HEAR ==> HEARING ==> HEART ==> HEARTWATER ==> HEARTWORM

and many thousands of others!

Syntax-related Problems

NL topical surrogates, e.g. book and journal titles, headlines, table of contents, back-of-the-book indexes, are usually expressed via larger syntactic units, mostly noun phrases.

Noun phrases are frequently ambiguous in and of themselves, e.g. SHELLFISH POISONING ==> POISONING OF SHELLFISH ==> POISONING BY SHELLFISH. The use of Boolean operators, metric

operators and a out of n weighted logic closest match search strategy in order to compensate for the lack of linguistic analysis in indexing, further compounds the text matching problem. Consider for example:

```
ABUSE ==> CHILD ABUSE ==> DRUG ABUSE ==>
          ELDER ABUSE ==> SPOUSE ABUSE

CRISIS MANAGEMENT ==> MANAGEMENT CRISIS ==> MANAGEMENT BY
          CRISIS ==>  CRISIS BY MANAGEMENT

KIDNEY ==> KIDNEY BEAN LECTINS ==>
          KIDNEY DISEASES

SEXUAL ABSTINENCE ==> SEXUAL PERVERSION

SHORT TERM EFFECTS ==> SHORT TERM MEMORY ==>
          SHORT TERM PSYCHOTHERAPY
```

Since literally hundreds of similar lexical and syntactic matching problems are encountered daily in a large operational NL IR system, it is evident that automatic query analysis and matching can substantially benefit from morphological and syntactic analysis in order to lend additional precision to the available truncation, character masking, Boolean, metric, weighted logic or generalized pattern matching strategies. [Consider for instance the automatic generation of Boolean search statements from NL queries (34).]

Semantic Problems

A great many formal semantic aids and ad hoc heuristics are used by trained searchers when interacting with IR systems. Some examples are controlled vocabularies, "hedges", "pre-explodes", multi database cross indices, stored search strategies and the like. Systems that rely on controlled vocabularies often lack in currency, database warrant or conceptual exhaustivity. For instance, the Medical Subject Headings thesaurus does not currently (1986 edition) have a subject heading for BIOTECHNOLOGY, and it uses PMS as a cross reference to an older subject heading PREGNANT MARE SYNDROME (GONADOTROPINS, EQUINE), but at the same time PMS is not linked to PREMENSTRUAL SYNDROME, nor is AI linked to ARTIFICIAL INTELLIGENCE !

As was noted earlier, systems without automated semantic aids shift the full burden of query understanding and matching on the searcher. NL IR interfaces must minimally rely on and must intelligently utilize existing machine readable semantic search aids. The existing aids, however, need to be augmented by additional semantic mapping tools, such as statistical term associations, switching vocabularies, enriched "fuzzy" thesauri, "scriptal" micro-lexicons, production rules, and/or heuristics elicited from expert searchers.

The NL IR interface must also be designed to deal with special problems, such as multiple languages, specialty languages, dialects and professional jargon (27, 35, 36).

To a considerable extent, the CITE experimental R & D system and its operational versions have attempted to address many of the linguistic problem areas outlined in this paper (6).

CITE represents a domain independent NL IR interface approach that combines conventional inverted file, Boolean logic with term frequency based weighted logic, closest match search strategy and efficient NLP techniques involving "intelligent" stemming (37), partial syntax analysis, automatic query-to-controlled-vocabulary mapping, lookahead search ambiguity resolution and filtering of combinatorial controlled vocabulary term displays, automatic user feedback processing and other techniques adapted from applied AI, such as selected domain specific semantic navigational tools, refined textual pattern matching and ad hoc expert searcher heuristics.

APPENDIX B illustrates several NL user interactions on the CITE system. The last example serves to put things in humbling perspective: the search query NATURAL LANGUAGE PROCESSING automatically picks up the subject heading NATURAL DISASTERS !!!

OTHER AI APPLICATIONS OF DIRECT RELEVANCE TO IR

In addition to NLP techniques in general and NL DBMS front ends in particular, the following AI areas are perceived by this author to be of direct relevance to IR:

1. Expert Systems

To the extent that rule based expert systems are instances of very high level programming tools that allow the expression of order independent rules instead of ad hoc pieces of order dependent conventional program code, they can be of benefit in NL IR interface development in capturing trained searcher expertise as well as codifying broad linguistic processing rules. Efficient microcomputer implementations of rule based expert systems are becoming increasingly available (7.6).

2. Intelligent Information Management

Intelligent information management involves the analysis of the interrelationships among multiple databases, information sources, user behavior, including observation of past actions and codified procedures, in order to develop rules for enhanced data retrieval and management (7.1). Elements of this approach have been utilized in IR, e.g. in the PAPERCHASE system (38) and the Syracuse University SUPARS Project. The systematic development and utilization of intelligent information management techniques should benefit IR systems in the future.

3. AI Knowledge Representation Techniques

In general, AI researcher have found that amassing large amounts of knowledge rather than sophisticated reasoning techniques, is responsible for the power of expert systems (7.7). The knowledge encoded in conventional controlled vocabularies can be potentially augmented via rule based relationships, "ISA" knowledge representation constructs (39) or predicate calculus statements and fuzzy logic.

4. INTEGRATING IR, DBMS, AI and OTHER TECHNOLOGIES

To date relatively little work has been done in such integrative R & D. Videotex (7.4) and CD-ROM systems are perhaps the most promising applications in this category. The latest videotex systems, e.g. the SOURCE and COMPUSERVE, combine sophisticated IR, DBMS, electronic mail and other technologies, and typically offer full text inversion and Boolean search, distributed database management as well as menu driven, tree structured user-friendly access. CD-ROM database publishing and special purpose knowledge base applications similarly combine state-of-the-art IR, DBMS, AI and overall information management technologies. The integration of efficient NLP techniques and intelligent computer assisted instruction (7.8) capabilities will enable videotex and CD-ROM users, and users of diverse information utilities to augment their own intellectual power by machine inteligence that is perhaps going to be able to grasp, if not understand database information content, discover new relationships, synthesize new knowledge and postulate new hypotheses.

S o m e d a y . . .

References

1. Salton, Gerard and McGill, Michael
Introduction to Modern Information Retrieval
McGraw-Hill Book Company, 1983, 448 pp.

2. Sparck-Jones, K. and Kay, M.
Linguistics and Information Science
Academic Press, N.Y., 1972, 344 pp.

3. Doszkocs, Tamas E.
"Searching MEDLINE in English: A Prototype User Interface with Natural Language Query, Ranked Output, and Relevance Feedback",
Proceedings of the ASIS, 14: 131-39, 1979.

4. Koll, M.; Noreault, T.; McGill, M.
"Enhanced Retrieval Techniques on a Microcomputer"
Proceedings of the National Online Meeting,
Learned Information, N.J., 169-178, 1984.

5. Bernstein, L.M. and Williamson, R.E.
"Testing of a Natural Language Retrieval System for a Full Text Knowledge Base"
JASIS, 35(4): 235-247, 1984.

6. Doszkocs, Tamas E.
"Natural Language Processing in Intelligent Information Retrieval",
Proceedings of the ACM Annual Meeting,
ACM, 1985, pp. 356-359.

7. Andriole, Stephen J., editor
Applications in Artificial Intelligence,
Petrocelli Books, Princeton, N.J., 1985, 528 pp.

7.1 -"- Schuster, Daniel
"Artificial Intelligence-Based Very Large Data Base Organization and Management", pp. 251-278.

7.2 -"- Hendrix, Gary G. and Sacerdoti, Earl D.
"Natural Language Processing: the Field in Perspective", pp. 149-192.

7.3 -"- Schank, Roger and Schwartz, Steven P.
"The Role of Knowledge Engineering in Natural Language Systems", pp. 193-212.

7.4 -"- Rice, Gerald F. and Andriole, Stephen J.
"Artificially Intelligent Videotex", pp. 295-312.

7.5 -"- Carbonell, Jaime G.
"The Role of User Modeling
in Natural Language", pp. 213-226.

7.6 -"- Lehner, Paul E. and Barth, Stephen W.
"Expert Systems on Microcomputers:, pp. 189-124.

7.7 -"- Gevarter, William B.
"Expert Systems: Limited but Powerful", pp.
125-142.

7.8 -"- Fletcher, J. Dexter
"Intelligent Instructional
Systems in Training" pp. 427-452.

8. Faloutsos, Christos
"Access Methods for Text"
Computing Surveys, 17(1): 49-74, March 1985.

9. Salton, Gerard , editor
The SMART Retrieval System: Experiments in
Automatic Document Processing,
Prentice Hall, 1971.

10. Bookstein, A.
"Implications of Boolean Structure for
Probabilistic Retrieval"
Proceedings of the Eighth Annual
International ACM SIGIR Conference,
Montreal, Canada, June 5-7, 1985,
pp. 11-17.

11. Kraft, D.H. and Bookstein, A.
"Evaluation of Information Retrieval Systems:
a Decision Theory Approach",
Journal of ASIS, 29: 31-40, 1978.

12. van Rijsbergen, C.J.
Information Retrieval, 2nd edition,
Butterworths, London, 1979.

13. Dosshocs, Tamas E.
"CITE NLM: Natural Language Searching in an
Online Catalog",
Information Technology and Libraries, 2(4):
364-380, 1983.

14. Lunin, L. and Smith, L., editors
"Perspectives on Artificial Intelligence:
Concepts, Techniques, Applications, Promise"
JASIS, 35(4): 277-315, 1984.

15. Grishman, R.
"Natural Language Processing"
JASIS, 35(4): 292-296, 1984.

16. Cooper, W.S.
"Bridging the Gap between AI and IR"
Research and Development in Information Retrieval,
Proceedings of the Third Joint BCS and ACM
Symposium, Cambridge, 1984, pp. 259-265.

17. Sparck-Jones, K.
"Natural Language Access to Databases:
Some Questions and a Specific Approach",
Journal of Information Science, 4(1):
41-48, 1982.

18. Kolodner, J.
"Indexing and Retrieval Strategies for
Natural Language Retrieval",
ACM Transactions on Database Systems,
8(3): 434-464, 1983.

19. Hayes-Roth, Frederick
"Rule-Based Systems",
Communications of the ACM, 28(9): 921-932, 1985.

20. Fikes, Richard and Kehler, Tom
"The Role of Frame-Based Representation in
Reasoning",
Communications of the ACM, 28(9): 904-920, 1985.

21. Politt, A.S.
"A 'front-end' system: an Expert System as an Online
Search Intermediary",
Aslib Proceedings, 36(5): 229-234, May 1984.

22. Brownston, Lee et al.,
Programming Expert Systems in OPS5,
Addison-Wesley, 1985, 471 pp.

23. Zampolli, A., editor,
Linguistic Structures Processing,
North-Holland, 1977, 585 pp.

23.1 -"- Kay, Martin
"Morphological and syntactic analysis",
pp. 131-234.

23.2 -"- Petrick, S.R.
"On Natural Language Based Computer Systems"
pp. 313-348.

24. Pacak, M.G. and Dunham, G.S.
"Computers and Medical Language",
Medical Informatics, 4(1): 13-27, 1979.

25. Tomita, Masaru
Efficient Parsing for Natural Language,
Kluwer Academic Publishers, 1986, 201 pp.

26. Golden, F.L.
Just what the doctor ordered. A recording of medical
humor,
New York, Pell, 1949, 256 pp.

27. Cowan, L. and Cowan, M.
The Wit of Medicine,
Frewin, 1972, 96 pp.

28. Charniak, Eugene and McDermott, Drew
Introduction to Artificial Intelligence,
Addison-Wesley, 1985, 781 pp.

29. Schmucker, Kurt J.
Fuzzy Sets, Natural Language Computations,
and Risk Analysis,
Computer Science Press, 1984, 192 pp.

30. Dosshocs, Tamas E.
"From Research to Application: the CITE Natural
Language Information Retrieval System",
In: Proceedings of the Fifth BCS and ACM SIGIR
Conference, Springer Verlag, Berlin, 1983,
pp. 251-262.

31. Rich, E.
"Natural Language Interfaces",
Computer, September 1984, pp. 39-47.

32. Dosshocs, Tamas E.
"An Associative Interactive Dictionary for Online
Bibliographic Searching",
Proceedings of the ASIS, 15: 105-109, 1978.

33. Dosshocs, Tamas E., Schultheiss, Robert J., and
Vasta, Bruno M.
"Analysis of Term Distribution in the TOXLINE
Inverted File"
Journal of Chemical Information and Computer
Sciences, 16: 131-135, 1976.

34. Salton, G., Buckley, C., Fox, E.
"Automatic Query Formulations in Information
Retrieval"
JASIS, 34(4): 262-280, 1983.

35. Bove, A. et al.
[Hellenic Influence in Medical English]
Med Clin, 83: 209-213, 1984. (Spanish)

36. Lyman, J. T.
"Dialect in Diagnostic",
Annals of Internal Medicine, 100: 899-901,
June 1984.

37. Ulmschneider, John E. and Dosshocs, Tamas E.
"A Practical Stemming Algorithm for Online
Search Assistance"
Online Review, 7(4): 301-318, 1983.

38. Borovits, G.L. and Bleich, H.L.
"Paperchase: A Computer Program to Search the
Medical Literature",
New England Journal of Medicine 305: 934-38, 1981.

39. Rada, Roy, Humphrey, Suzanne and Coccia, Craig
"A knowledge Base for Retrieval Evaluation",
Proceedings of ACM 85: 360-367,
ACM, October 1985.

APPENDIX A

Example of "wretched" and "incomprehensible" natural language prose.

SATURDAY, JANUARY 18, 1986

The Washington Post

What Does It Mean? Condo Owner Asks

By R.H. Melton
Washington Post Staff Writer

ANNAPOLIS—When Mortimer D. Goldstein wanted to show his students at the National Defense College an example of wretched and confusing prose, he read a 307-word sentence contained in the bylaws of his own condominium.

Simply put, one of the condominium bylaws at Leisure World, the huge retirement community in Montgomery County, says: Make no alterations to the exterior of your condominium or any structural changes to the interior.

What offended Goldstein's respect for clarity in English was that the message is delivered in a torrent of words, rather than in short, declarative sentences. Goldstein argued that he and other legal laymen could barely make it past the first comma, nevermind understand the full meaning of the important condominium rule.

Goldstein, a former State De-

See CONDOS, F8, Col. 1

From the Bylaws. Any Questions?

Here is an excerpt from the bylaws of one of the condominiums at the Leisure World community in Silver Spring. This sentence contains 307 words. It was written by the developer's lawyer.

Section 1. Architectural and Environmental Control Committee. Except for the construction of the condominium by the Declarant or its agents and any improvement to any condominium unit or to the common elements accomplished concurrently with said original construction, and except for purposes of proper maintenance and repair or as otherwise in the Condominium Act or these By-Laws provided, it shall be prohibited for any unit owner to install, erect, attach, apply, paste, hinge, screw, nail, build, alter, remove or construct any lighting, shades, screens, awnings, patio covers, decorations, fences, walks, aerials, antennas, radio or television broadcasting or receiving devices, slabs, sidewalks, curbs, gutters, patios, balconies, porches, driveways, walls or to make any change or otherwise alter (including any alteration in color) in any manner whatsoever the exterior of any condominium unit or upon any of the common elements within the project or to combine or otherwise join two or more condominium units, or to partition the same after combination, or to remove or alter any window or exterior doors of any condominium unit, or to make any change or alteration within any condominium unit which will alter the structural integrity of any building or otherwise affect the property interest or welfare of any other unit owner, materially increase the cost of operating or insuring the condominium or impair any easement, until the complete plans and specifications, showing the location, nature, shape, change (including, without limitation, any other information specified by the Board of Directors or its designated committee) shall have been submitted to and approved in writing as to safety, the effect of any such alterations on the costs of maintaining and insuring the condominium and harmony of design, color and location in relation to surrounding structures and topography, by the Board of Directors of the Council of Unit Owners, or by an Architectural and Environmental Control Committee designated by the Board of Directors.

APPENDIX B

NATURAL LANGUAGE SEARCH EXAMPLES

ON THE CITE SYSTEM

```
SELECT ( 1=SUBJECT, 2=AUTHOR, 3=OTHER FILE, 4=STOP ) :1

SUBJECT SEARCH - PLEASE TYPE YOUR SEARCH TOPIC
:symptoms of post traumatic stress disorder among vietnam veterans

 Looking  for terms to use in searching...

<VIETNAM:> NOT FOUND

Try different spelling, or type HELP, or press the RETURN key to proceed
:vietnam

 CHOOSE FROM THE FOLLOWING 11 MEDICAL SUBJECT HEADINGS (MH), TEXT WORDS (TW) OR
SUBHEADINGS (SH) :

 CHOICE    TERM                                                  NUMBER OF ITEMS
 ------    ----                                                  ---------------
   1  STRESS DISORDERS, POST-TRAUMATIC  ( mh )                             184
   2  VIETNAM  ( mh )                                                      184
   3  VIETNAM  ( tw )                                                      182
   4  WOUNDS AND INJURIES  ( mh )                                         1244
   5  VETERANS  ( tw )                                                     254
   6  TRAUMATIC  ( tw )                                                   1782
   7  STRESS  ( mh )                                                      1206
   8  DISORDERS  ( tw )                                                   4027
   9  STRESS  ( tw )                                                      5209
  10  POST  ( tw )                                                        7603
  11  SYNDROME  ( tw )                                                    9761

TYPE THE CHOICE NUMBERS OF THE SEARCH TERMS YOU WANT TO USE
 IN THEIR ORDER OF IMPORTANCE  or type ALL or NONE

:11 and 1-6

 Search in progress...

      505,060 ITEMS SEARCHED

       946 ITEMS CONTAIN ONE OR MORE OF YOUR SEARCH TERMS

        10  ITEMS  MATCH YOUR SEARCH TOPIC CLOSELY

WOULD YOU LIKE TO SEE ONLY ENGLISH LANGUAGE ITEMS? Y/N:Y



THE RETRIEVED ITEMS CONTAIN THE FOLLOWING TERM COMBINATIONS ...

CHOICE  NUMBER OF    SEARCH TERMS THAT LED TO
NUMBER    ITEMS          THESE ITEMS
------  --------  ------------------------------
   1        10     STRESS (TW); WOUNDS AND INJURIES (MH);
                   STRESS DISORDERS, POST-TRAUMATIC (MH);
                   VETERANS (MH); VIETNAM (TW)
   2         1     STRESS (TW);
                   STRESS DISORDERS, POST-TRAUMATIC (MH);
                   VIETNAM (TW)
   3        15     STRESS (TW);
                   STRESS DISORDERS, POST-TRAUMATIC (MH)
   4         2     WOUNDS AND INJURIES (MH);
                   STRESS DISORDERS, POST-TRAUMATIC (MH);
                   VETERANS (MH); VIETNAM (TW)
   5        14     STRESS (TW); WOUNDS AND INJURIES (MH)
   6         6     STRESS DISORDERS, POST-TRAUMATIC (MH)
   7         1     STRESS (TW); VETERANS (MH)
   8         3     DISTRESS (TW); VIETNAM (TW)

Press the RETURN key to continue display,
(or type S <select other options>, or X <new search> or STOP):S


Type CHOICE NUMBER(S) or NUMBER RANGE(S) or ALL or NONE
 or type X <new search>, or type STOP

:1,3,6


   1/; Lawfer MB; Brett E; Gallops MB; Symptom patterns associated with posttraumatic stress disorder among Vietnam veterans exposed to war trauma.; Am J Psychiatry; 1985 Nov; 142; 11; 1304-11; Eng

   2/; Van Dyke C; Zilberg NJ; Goldstone JA; Posttraumatic stress disorder: a thirty-year delay in a World War II veteran.; Am J Psychiatry; 1983 Sep; 140; 9; 1172-3; Eng

   3/; Jelinek JM; Williams V; Post-Traumatic Stress Disorder and substance abuse in Vietnam combat veterans: treatment problems, principles and recommendations.; J Subst Abuse Treat; 1984; 1; 3; 27-37; Eng

Press the RETURN key to continue display,
```

```
SUBJECT SEARCH - PLEASE TYPE YOUR SEARCH TOPIC
:anything on vitamin b deficiency in children

 Looking  for terms to use in searching...

 CHOOSE FROM THE FOLLOWING 8 MEDICAL SUBJECT HEADINGS (MH), TEXT WORDS (TW) OR
SUBHEADINGS (SH) :

 CHOICE    TERM                                                  NUMBER OF ITEMS
 ------    ----                                                  ---------------
   1  VITAMIN B DEFICIENCY  ( mh )                                         23
   2  PEDIATRICS  ( mh )                                                  583
   3  PEDIATRICS  ( tw )                                                  598
   4  VITAMIN  ( tw )                                                    3995
   5  CHILD  ( mh )                                                     36698
   6  DEFICIENCY  ( tw )                                                 5329
   7  CHILDREN  ( tw )                                                  15492
   8  B  ( tw )                                                         19748

TYPE THE CHOICE NUMBERS OF THE SEARCH TERMS YOU WANT TO USE
 IN THEIR ORDER OF IMPORTANCE  or type ALL or NONE

:1 and 5-7

 Search in progress...

      505,060 ITEMS SEARCHED

      1,585 ITEMS CONTAIN ONE OR MORE OF YOUR SEARCH TERMS

          5  ITEMS  MATCH YOUR SEARCH TOPIC CLOSELY

WOULD YOU LIKE TO SEE ONLY ENGLISH LANGUAGE ITEMS? Y/N:Y

Type the  LANGUAGE(S) of your choice or ALL:all

 THE RETRIEVED ITEMS CONTAIN THE FOLLOWING TERM COMBINATIONS ...

CHOICE  NUMBER OF    SEARCH TERMS THAT LED TO
NUMBER    ITEMS          THESE ITEMS
------  --------  ------------------------------
   1         5     CHILD (MH); CHILDREN (TW); DEFICIENCY (TW);
                   VITAMIN B DEFICIENCY (MH)
   2        15     DEFICIENCY (TW); VITAMIN B DEFICIENCY (MH)
   3       615     CHILD (MH); CHILDREN (TW); DEFICIENCY (TW)
   4       113     CHILDREN (TW); DEFICIENCY (TW)
   5       838     DEFICIENCY (TW)

Type CHOICE NUMBER(S) or NUMBER RANGE(S) or ALL or NONE
 or type X <new search>, or type STOP


:1


   1/; Hamamoto CD; Infectious oral sarcosis (cancrum oris) in Nigerian children: a review.; Community Dent Oral Epidemiol; 1985 Jun; 13; 3; 190-4; En

   2/; Hennart HR; Nutrition for the practitioner V. Current concepts in nutrition: the vitamin B complex.; J Ala State Med Assoc; 1984 Nov; 77; 11; 853-6; Eng

   3/; Salvesol HK; How to mobilize a rural community for a health programme.; Oygie; 1984 Sept; 3; 9-12; Eng

Press the RETURN key to continue display,
(or type S <select other options>, or X <new search> or STOP):


   4/; Shin TS; Rosenberg R; Endres W; Pyridoxal-5'-phosphate concentration as carrier for vitamin-B6-dependent seizures in the newborn [letter]; Lancet; 1984 Oct 13; 2; 0407; 879-1; Eng

   5/; Berger H; [Wholesome nutrition]; Wien Med Wochenschr; 1984 Apr 30; 134; 8; 183-5; Ger

 ALL SELECTED ITEMS HAVE BEEN DISPLAYED

 SELECT FROM THE FOLLOWING OPTIONS:

       1 = DISPLAY IN BRIEF THE ITEM(S) OF YOUR CHOICE
       2 = DISPLAY IN DETAIL THE ITEM(S) OF YOUR CHOICE
       3 = FIND ITEMS SIMILAR TO THE ONE(S) OF YOUR CHOICE
       4 = SHOW HOW MANY ITEMS CONTAIN EACH
           OF YOUR SEARCH TERMS
       5 = START NEW SEARCH
       6 = STOP SEARCHING
```

```
oooooooooooooooooooooooooooooooooooooooooooo
o type words, phrases or full topic and press the RETURN key o
oooooooooooooooooooooooooooooooooooooooooooo

SUBJECT SEARCH - PLEASE TYPE YOUR SEARCH TOPIC

:the use of biofeedback for the treatment of tension headache and migraine

 Looking  for terms to see in searching...

 CHOOSE FROM THE FOLLOWING 10 MEDICAL SUBJECT HEADINGS (MH), TEXT WORDS (TW) O

SUBHEADINGS (SH) :

 CHOICE   TERM                                                    NUMBER OF ITEMS
 ------   ----                                                    ---------------
  1  BIOFEEDBACK (PSYCHOLOGY)  ( sh )                                        314
  2  BIOFEEDBACK  ( tw )                                                     179
  3  MIGRAINE  ( sh )                                                        544
  4  MIGRAINE  ( tw )                                                        425
  5  HEADACHE  ( sh )                                                        498
  6  HEADACHE  ( tw )                                                        913
  7  BIOQUEST  ( sh )                                                        510
  8  TENSION  ( tw )                                                        2189
  9  DRUG THERAPY  ( sh )                                                    487
 10  BIOQUEST  ( sh )                                                      30868
 11  BIOQUEST  ( tw )                                                      13468
 12  DRUG EFFECTS  ( sh )                                                  69131
 13  DRUG THERAPY  ( sh )                                                  39622
 14  DRUG  ( tw )                                                          17269
 15  THERAPY  ( tw )                                                       22859

Press the RETURN key to continue display,
(or type S <select other options>, or X <new search> or STOP):

 16  THERAPEUTIC USE  ( sh )                                               44769
 17  THERAPY  ( sh )                                                       31969
 18  TREATMENT  ( tw )                                                     55695

TYPE THE CHOICE NUMBERS OF THE SEARCH TERMS YOU WANT TO USE
 IN THEIR ORDER OF IMPORTANCE  or type ALL or NONE

:6, 5 4 1 thru 4 and 15
```

```
 Search in progress...

      585,844 ITEMS SEARCHED

     1,373 ITEMS CONTAIN ONE OR MORE OF YOUR SEARCH TERMS

        1  ITEM MATCHES  YOUR SEARCH TOPIC CLOSELY

WOULD YOU LIKE TO SEE ONLY ENGLISH LANGUAGE ITEMS? Y/N:no

Type the  LANGUAGE(s) of your choice or ALL:all

 THE RETRIEVED ITEMS CONTAIN THE FOLLOWING TERM COMBINATIONS ...

 CHOICE  NUMBER OF     SEARCH TERMS THAT LED TO
 NUMBER    ITEMS          THESE ITEMS
 ------  ---------     ------------------------

    1          1     THERAPY (TW); TENSION (TW); HEADACHE (TW);
                      MIGRAINE (MH); HEADACHE (MH); MIGRAINE (TW);
                      BIOFEEDBACK (PSYCHOLOGY) (MH); BIOFEEDBACK (TW)
    2          1     TENSION (TW); HEADACHE (TW); MIGRAINE (MH);
                      HEADACHE (MH); MIGRAINE (TW);
                      BIOFEEDBACK (PSYCHOLOGY) (MH); BIOFEEDBACK (TW)
    3          1     THERAPY (TW); TENSION (TW); HEADACHE (TW);
                      HEADACHE (MH); BIOFEEDBACK (PSYCHOLOGY) (MH);
                      BIOFEEDBACK (TW)
    4          1     THERAPY (TW); TENSION (TW); HEADACHE (TW);
                      MIGRAINE (MH); HEADACHE (MH); MIGRAINE (TW)
    5          6     TENSION (TW); HEADACHE (TW); HEADACHE (MH);
                      BIOFEEDBACK (PSYCHOLOGY) (MH); BIOFEEDBACK (TW)
    6         22     TENSION (TW); HEADACHE (TW); MIGRAINE (MH);
                      HEADACHE (MH); MIGRAINE (TW)
    7          2     THERAPY (TW); TENSION (TW); HEADACHE (TW);
                      HEADACHE (MH)
    8          1     THERAPY (TW); TENSION (TW);
                      BIOFEEDBACK (PSYCHOLOGY) (MH); BIOFEEDBACK (TW)

Press the RETURN key to continue display,
(or type S <select other options>, or X <new search> or STOP):S


Type CHOICE NUMBER(s) or NUMBER RANGE(s) or ALL or NONE
 or type X <new search>, or type STOP

:1-5
```

1/; Woolley-Hart A; [Biofeedback and relaxation therapy for patients with migraine and tension headaches]; Psychiatr Neurol Med Psychol (Leipz); 1984 Nov; 36; 11; 649-54; Ger

2/; Appelboom EA; Blanchard EB; Andrasik F; Muscle discrimination ability at three muscle sites in three headache groups.; Biofeedback Self Regul; 1984 Dec; 9; 4; 421-38; Eng

3/; Blanchard EB; Andrasik F; Appelboom EA; Evans DD; Jurish SE; Taders SJ; Hedlund LD; Barron KD; The efficacy and cost-effectiveness of minimal-therapist-contact, non-drug treatments of chronic migraine and tension headache.; Headache; 1985 Jun; 25; 4; 214-38; Eng

Press the RETURN key to continue display,
(or type S <select other options>, or X <new search> or STOP):

4/; Kieg AC; Arena JG; Behavioral treatment of chronic cluster headache in a geriatric patient.; Biofeedback Self Regul; 1984 Jun; 9; 2; 281-9; Eng

5/; Veitl D; ["One need not be a fakir or guru."; Biofeedback as an interdisciplinary research technic]; Fortschr Med; 1984 May 17; 102; 19; 541-4; Ger

6/; Doerr-Proske B; Wittchen HU; [A muscle and vascular oriented relaxation program for the treatment of chronic migraine patients. A randomized clinical comparative study]; Z Psychosom Med Psychoanal; 1985; 31; 3; 247-66; Ger

Press the RETURN key to continue display,
(or type S <select other options>, or X <new search> or STOP):S

```
 SELECT FROM THE FOLLOWING OPTIONS:

       1 = DISPLAY IN BRIEF THE ITEM(S) OF YOUR CHOICE
       2 = DISPLAY IN DETAIL THE ITEM(S) OF YOUR CHOICE
       3 = FIND ITEMS SIMILAR TO THE ONE(S) OF YOUR CHOICE
       4 = SHOW HOW MANY ITEMS CONTAIN EACH
           OF YOUR SEARCH TERMS
       5 = START NEW SEARCH
       6 = STOP SEARCHING

:2


CHOOSE THE ITEM(S) IN WHICH YOU ARE INTERESTED

Type CHOICE NUMBER(s) or NUMBER RANGE(s) or ALL or NONE
 or type X <new search>, or type STOP

:1 and 4
```

1/; Woolley-Hart A; [Biofeedback and relaxation therapy for patients with migraine and tension headaches]; Psychiatr Neurol Med Psychol (Leipz); 1984 Nov; 36; 649-54; Ger; The biofeedback literature shows that biofeedback training is most effective when combined with relaxation and visualization techniques. This approach was used to run a pilot study for training 69 people with migraine and tension headache. Our results indicate an improvement in both headache symptomology, general well-being and a reduction in anxiety. In addition, our observations from working with these patients suggests that the training programme could be improved by adding physical measures such as the re-education of breathing patterns and posture. We also found that insight, awareness of factors causing stress and an ability to change life style were an essential ingredient to reduction of headache intensity of frequency.; Adolescence; Adult; Anxiety; Biofeedback (Psychology); Combined Modality Therapy; English Abstract; Female; Follow-Up Studies; Headache; Human; Male; Middle Age; Migraine; Relaxation Technics; Support, Non-U.S. Gov't

4/; Kieg AC; Arena JG; Behavioral treatment of chronic cluster headache in a geriatric patient.; Biofeedback Self Regul; 1984 Jun; 9; 281-9; Eng; While nonpharmacological treatment of migraine and tension headache has increasingly been demonstrated to be efficacious, relatively little attention has been focused upon treatment of the more severe cluster headache. This has particularly been the case for the type of cluster headache known as chronic cluster. The present case study focused upon the treatment of a 69-year-old male with a 37-year history of headache activity. The description of headache satisfied those criteria currently considered to be indicative of chronic cluster headache. Following 6 weeks of baseline, during which daily ratings of head pain and daily ingestion of "as needed" (PRN) pain medication were collected, a 7-week treatment phase was implemented. Treatment consisted of thermal biofeedback coupled with a spouse contingency program (i.e., directing the spouse to avoid consequating subject reports of head pain). During the treatment phase, a near-100% reduction in amount of weekly PRN pain medication ingested was noted, along with a decrease in self-reports of head pain. Both of these decreases were maintained at 1-month, 6-month, and 15-month follow-ups. Implications for treatment of chronic cluster headache were discussed.; Aged; Biofeedback (Psychology); Case Report; Cluster Headache; Human; Male; Skin Temperature; Support, U.S. Gov't, Non-P.H.S.; Vascular Headache

```
 SELECT FROM THE FOLLOWING OPTIONS:

       1 = DISPLAY IN BRIEF THE ITEM(S) OF YOUR CHOICE
       2 = DISPLAY IN DETAIL THE ITEM(S) OF YOUR CHOICE
       3 = FIND ITEMS SIMILAR TO THE ONE(S) OF YOUR CHOICE
       4 = SHOW HOW MANY ITEMS CONTAIN EACH
           OF YOUR SEARCH TERMS
       5 = START NEW SEARCH
       6 = STOP SEARCHING
```

```
SELECT ( 1=SUBJECT, 2=AUTHOR, 3=OTHER FILE, 4=STOP ) :1

SUBJECT SEARCH - PLEASE TYPE YOUR SEARCH TOPIC

:genetic screening for malignant hyperthermia disease susceptibility

 Looking  for terms to use in searching...

 CHOOSE FROM THE FOLLOWING 11 MEDICAL SUBJECT HEADINGS (MH), TEXT WORDS (TW) OR
SUBHEADINGS (SH) :

 CHOICE    TERM                                                  NUMBER OF ITEMS
 ------    ----                                                  ---------------
   1  MALIGNANT HYPERTHERMIA  ( mh )                                         174
   2  HYPERTHERMIA  ( tw )                                                   833
   3  GENETIC SCREENING  ( mh )                                               75
   4  DISEASE SUSCEPTIBILITY  ( mh )                                         817
   5  SUSCEPTIBILITY  ( tw )                                                2861
   6  SCREENING  ( tw )                                                     4949
   7  GENETICS  ( mh )                                                     21918
   8  GENETIC  ( tw )                                                       5946
   9  MALIGNANT  ( tw )                                                     7438
  10  PATHOLOGY  ( mh )                                                    49935
  11  DISEASE  ( tw )                                                      35167

TYPE THE CHOICE NUMBERS OF THE SEARCH TERMS YOU WANT TO USE
 IN THEIR ORDER OF IMPORTANCE  or type ALL or NONE

:1 2 4

 Search in progress...

      585,864 ITEMS SEARCHED

     1,815 ITEMS CONTAIN ONE OR MORE OF YOUR SEARCH TERMS

         3  ITEMS  MATCH YOUR SEARCH TOPIC CLOSELY

WOULD YOU LIKE TO SEE ONLY ENGLISH LANGUAGE ITEMS? Y/N:

THE RETRIEVED ITEMS CONTAIN THE FOLLOWING TERM COMBINATIONS ...

 CHOICE    NUMBER OF      SEARCH TERMS THAT LED TO
 NUMBER      ITEMS             THESE ITEMS
 ------    ---------      ----------------------
   1           3      DISEASE SUSCEPTIBILITY (MH);
                       MALIGNANT HYPERTHERMIA (MH);
                       GENETIC SCREENING (MH)
   2          15      DISEASE SUSCEPTIBILITY (MH);
                       MALIGNANT HYPERTHERMIA (MH)
   3         154      MALIGNANT HYPERTHERMIA (MH)
   4           2      DISEASE SUSCEPTIBILITY (MH);
                       GENETIC SCREENING (MH)
   5          78      GENETIC SCREENING (MH)
   6         769      DISEASE SUSCEPTIBILITY (MH)



Type CHOICE NUMBER(s) or NUMBER RANGE(s) or ALL or NONE
 or type X (new search), or type STOP

:1, 3


   1/: A protocol for the investigation of malignant hyperpyrexia (MH) susceptibility. The European Malignant Hyperpyrexia Group.; Br J Anaesth; 1984 Nov; 56; 11; 1267-9; Eng

   2/: Ording H; Brandom B; Fletcher R; Investigation of malignant hyperthermia in Denmark and Sweden.; Br J Anaesth; 1984 Nov; 56; 11; 1193-99; Eng

   3/: Ellis FR; European Malignant Hyperpyrexia Group [editorial]; Br J Anaesth; 1984 Nov; 56; 11; 1161-2; Eng

Press the RETURN key to continue display,
(or type S (select other options), or X (new search) or STOP):


   4/: Rosenberg H; Malignant hyperthermia.; Hosp Pract (Off); 1985 Mar 15; 20; 3; 139, 144-5, 148 passim; Eng

   5/: Ilias WK; Williams CB; Fulfer MT; Denier SE; Diltiazem inhibits halothane-induced contractions in malignant hyperthermia-susceptible muscles in vitro.; Br J Anaesth; 1985 Oct; 57; 10; 994-6; Eng

   6/: Ranklev E; Ansati H; Fletcher R; Microcalorimetric studies in malignant hyperpyrexia susceptible individuals.; Br J Anaesth; 1985 Oct; 57; 10; 991-3; Eng

Press the RETURN key to continue display,
(or type S (select other options), or X (new search) or STOP):x
```

```
SELECT ( 1=SUBJECT, 2=AUTHOR, 3=OTHER FILE, 4=STOP ) :1

SUBJECT SEARCH - PLEASE TYPE YOUR SEARCH TOPIC

:natural language processing in information retrieval

 Looking  for terms to use in searching...

 CHOOSE FROM THE FOLLOWING  9 MEDICAL SUBJECT HEADINGS (MH), TEXT WORDS (TW) OR
SUBHEADINGS (SH) :

 CHOICE    TERM                                                  NUMBER OF ITEMS
 ------    ----                                                  ---------------
   1  INFORMATION SYSTEMS  ( mh )                                            659
   2  RETRIEVAL  ( tw )                                                      337
   3  LANGUAGE  ( mh )                                                       368
   4  LANGUAGE  ( tw )                                                       894
   5  AUTOMATIC DATA PROCESSING  ( mh )                                      129
   6  PROCESSING  ( tw )                                                    3264
   7  NATURAL DISASTERS  ( mh )                                               35
   8  NATURAL  ( tw )                                                       5331
   9  INFORMATION  ( tw )                                                   8299

TYPE THE CHOICE NUMBERS OF THE SEARCH TERMS YOU WANT TO USE
 IN THEIR ORDER OF IMPORTANCE  or type ALL or NONE

:8 3 4 1 2

 Search in progress...

      585,864 ITEMS SEARCHED

     1,747 ITEMS CONTAIN ONE OR MORE OF YOUR SEARCH TERMS

     NONE OF THE ITEMS MATCH ALL THE SEARCH SETS EXACTLY


WOULD YOU LIKE TO SEE ONLY ENGLISH LANGUAGE ITEMS? Y/N:yes

THE RETRIEVED ITEMS CONTAIN THE FOLLOWING TERM COMBINATIONS ...

 CHOICE    NUMBER OF      SEARCH TERMS THAT LED TO
 NUMBER      ITEMS             THESE ITEMS
 ------    ---------      ----------------------
   1           6      NATURAL (TW); LANGUAGE (TW);
                       INFORMATION SYSTEMS (MH); RETRIEVAL (TW)
   2           7      NATURAL (TW); LANGUAGE (TW); LANGUAGE (MH)
   3           6      NATURAL (TW); INFORMATION SYSTEMS (MH);
                       RETRIEVAL (TW)
   4           1      NATURAL (TW); LANGUAGE (TW); RETRIEVAL (TW)
   5           1      LANGUAGE (TW); INFORMATION SYSTEMS (MH);
                       RETRIEVAL (TW)
   6          15      NATURAL (TW); LANGUAGE (TW)
   7          69      LANGUAGE (TW); LANGUAGE (MH)
   8           2      NATURAL (TW); RETRIEVAL (TW)

Press the RETURN key to continue display,
(or type S (select other options), or X (new search) or STOP):

   9         116      INFORMATION SYSTEMS (MH); RETRIEVAL (TW)
  10       1,127      NATURAL (TW)
  11         123      LANGUAGE (TW)
  12          78      RETRIEVAL (TW)

Type CHOICE NUMBER(s) or NUMBER RANGE(s) or ALL or NONE
 or type X (new search), or type STOP

:1 and 4


   1/: Roach J; Lee S; Wilcox J; Ehrich R; An expert system for information on pharmacology and drug interactions.; Comput Biol Med; 1993; 23; 1; 11-23; Eng

   2/: Pocsko PJ; Ackerman LV; Galpert MD; Simms SM; Developing an integrated natural language database for gastrointestinal disease.; Henry Ford Hosp Med J; 1984; 32; 3; 178-3; Eng

   3/: Gupta SK; Veith FJ; White-Flores SA; Samson RH; Scher LA; Weiser MI; Aaron S; System for widespread application of microcomputers to vascular surgery.; J Vasc Surg; 1984 Jul; 1; 4; 691-4; Eng

Press the RETURN key to continue display,
(or type S (select other options), or X (new search) or STOP):Y


   4/: Pocsko PJ; Ackerman LV; Galpert MD; Simms SM; A computer-based gastrointestinal information management system.; Radiology; 1984 Aug; 153; 2; 297-300; Eng

  16/: Moore GW; Hutchins GM; Miller RE; Strategies for searching medical natural language text. Distribution of words in the anatomic diagnoses of 7000 autopsy subjects.; Am J Pathol; 1984 Apr; 115; 1; 36-41; Eng

 ALL SELECTED ITEMS HAVE BEEN DISPLAYED

 SELECT FROM THE FOLLOWING OPTIONS:


       1 = DISPLAY IN BRIEF THE ITEM(S) OF YOUR CHOICE
       2 = DISPLAY IN DETAIL THE ITEM(S) OF YOUR CHOICE
       3 = FIND ITEMS SIMILAR TO THE ONE(S) OF YOUR CHOICE
       4 = SHOW HOW MANY ITEMS CONTAIN EACH
           OF YOUR SEARCH TERMS
       5 = START NEW SEARCH
       6 = STOP SEARCHING

:
VIAS-13:03:32 PM. CPU-00:00:39 SERVICE-748 SESSION-00:10:40 JANUARY 20,1986

READY
```

THE VISUAL DISPLAY OF INFORMATION IN AN INFORMATION RETRIEVAL ENVIRONMENT

Donald B. Crouch
Department of Computer Science
Tulane University
New Orleans, LA 70118 USA

Abstract

This paper gives an overview of the graphical techniques which have been used in the representation of information in a document collection environment. An assessment of the applicability of existing multivariate data graphical techniques to the vector space model is presented.

1. Introduction

Computer graphics technology has advanced rapidly during the past few years. However, the graphics technology has not been fully incorporated into existing sophisticated information retrieval systems. Research in the application of graphics technology to on-line document retrieval has generally been limited to the enhancement of user-system interaction through the utilization of window management techniques and man-machine communication devices (such as the mouse). This type of human factors research is primarily concerned with system functioning and convenience as it relates to the user; its goal is to facilitate *effortless* learning and use of a system through the proper matching of graphics technology to user capabilities and limitations. [LIND85] The incentive for research focusing on optimization of the interaction between systems and users is largely economic. The cost structure of information processing has changed to the point that the "time of a person using an information system is often more costly than that of the system itself." [NICK84]

This study was supported in part by the National Science Foundation under grant IST 84-05740.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.
© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

Coinciding with the development of graphics technology has been a redirection of emphasis in the processing of multivariate statistical data from a purely mathematical analysis to one of graphical examination. [EVER78, WANG78, APPL81, BARN81, MYER81, HEIN83, BARN84] Graphical representation of data has long been regarded by statisticians as an extremely useful technique, and most now regard it as an essential prerequisite to a more formal analysis of the data. Emphasis of such techniques is on the review, display and manipulation of multivariate data as a means by which one may readily assimilate its underlying structure. In this regard, graphics is viewed as an instrument for reasoning about quantitative data. [TUFT83] Since document collections may be represented as multidimensional vector spaces, it is also appropriate to investigate the application of this aspect of graphics technology to the document retrieval process as a means of increasing retrieval effectiveness.

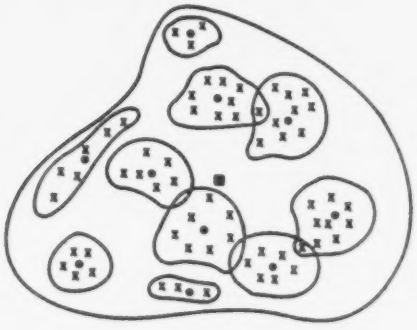
The purpose of this paper is to familiarize researchers with the approaches that have been taken in the graphical representation and manipulation of the document space and introduce the reader to the direction in which this research is now being focused.

2. The Document Space

In many information retrieval systems, the content of each document within the collection at large is represented by a set of content terms. Each document may then be considered to be a *term vector*, and the complete document collection becomes a *vector space* whose dimension is equal to the number of distinct terms used to identify the documents in the collection. [SALT68, VANR79] Similar or related documents are represented in such a collection by similar multidimensional term vectors. A vector representation of documents facilitates certain storage and retrieval operations: namely, (1) the construction of a clustered document file (consisting of classes of documents such that documents within a given class exhibit substantial similarities with each other); in clustered collections the search of a collection can

be limited to the documents within those clusters whose class vector representations are similar to the *query vector*, (2) the ranking of retrieved documents in decreasing order of their similarity with the query; the similarity of term vectors is often computed by using a correlation measure such as the cosine correlation, and (3) the reformulation of the query based on relevance assessments supplied by the user for previously retrieved documents. [SALT83]

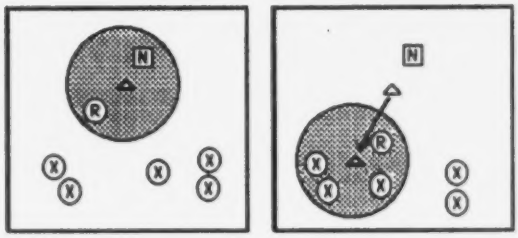
The concepts of clustering and query reformulation have been effectively illustrated using graphics art. For example, document relationships may be graphically visualized by representing term vectors as points in the vector space where the distance between points is inversely related to vector similarity; that is, two points will appear close to each other when their vectors are similar. A typical example of a document vector space appears in Fig. 1, where documents that are sufficiently similar have been included in a common cluster. [SALT78]



X - Stored Record
• - Cluster Centroid
■ - Centroid of Record Space

Fig. 1 Typical Clustered File Organization

Similarly, the query negotiation process may be represented graphically to illustrate how a reformulated query may actually move through the document space to new document areas. For example, the well known relevance feedback procedure may be viewed graphically as shown in Fig. 2. The initial query represented by Δ in Fig. 2 (a) first retrieves one nonrelevant (N) item and one relevant (R) item; when information from the relevant document is incorporated into the query, the query *migrates* downward as shown in Fig. 2 (b) where new documents, not previously seen, are now retrievable. [SALT83] One can envision the manner in which such displays could be used in an interactive retrieval environment if systems existed which could provide a visual representation of the document space.



(a) Initial Query (b) Reformulated Query

Fig. 2 Relevance Feedback Illustration

In document retrieval the data being manipulated may exhibit rather large dimensionality. A cursory review of the literature reveals that most statistically applied graphical techniques deal only with data of two or three dimensions. Yet, while many of these methods are applicable to higher dimensions, they often produce displays which are not particularly meaningful and which therefore convey little additional information about the data. In fact, some graphical approaches when extended to higher dimensions either "fail to improve the communication of the information or, worse still, misinform." [HEFF85] The discussion is therefore limited to those techniques which have either been used in a document environment, or appear to hold promise for such applications.

Two general approaches have emerged from studies dealing with the graphical representation of multidimensional data. Since a visual display of two-dimensional data may be easily produced, several analytical techniques have been developed which seek to obtain a representation of the original n-dimensional data in two dimensions. These so-called *ordination* or *mapping* techniques retain some of the data relationships present within the original multivariate data space. However the mapping processes involve extensive manipulations of the multidimensional data, and the displays may conceal the contribution of the individual vector components to the visual data display. Other graphical approaches have therefore emerged that represent the data points as icons, and the vector components (dimensions) as features of the icon. These methods are known as multivariate point graphs and are predicated on the assumption that the psychological perception of the icons will follow closely the relations of the points in vector space. [WILK82] The applicability of these two distinct graphical techniques to the representation of the document space is reviewed in the remainder of this study.

3. Correlation Mapping Techniques

Since a visual examination of multivariate data may be easily made when the data are functions of only two variables, mapping techniques were designed to translate the original multivariate data points to two-dimensional representations in a manner that approximately preserves the original vector associations. These methods are commonly referred to as *multidimensional scaling* (MDS) methods. Two primary MDS methods exist: metric and nonmetric. The objective of the metric MDS technique is to produce a coordinate representation in which the interpoint distances are approximately the same as the distances between corresponding points in the original space. Nonmetric MDS methods attempt to transform the data in a way that preserves the ranking of the points based on computed distances in the reduced space compared with the ranking of the original points based on pairwise similarities (thus the basis for the designation of these types of methods as *ordination* techniques). Early work in the nonmetric MDS area, resulting in the development of computer algorithms for these transformations, is described in [SHEP62a, SHEP62b, KRUS64a, KRUS64b].

Since MDS techniques generally operate directly on an n x n matrix of pairwise distances or similarities obtained from the n vectors in the original space, we briefly describe the means by which direct graphical displays of the similarity matrices themselves have been employed as a clustering aid. We then explore the application of the mapping techniques to document spaces.

3.1 Direct Visual Displays of Matrices

Several researchers have suggested methods for directly displaying similarity matrices which involve the substitution of symbols with varying shades of gray for the numerical elements in the matrix, high similarities or small distances being indicated by darker shades. These displays prove useful when the matrix is reconfigured so that the darker elements lie close to the diagonal. With this type of arrangement, clusters present in the original data become visually apparent.

A difficulty with this technique occurs when the matrix to be reordered is large, since no single ordering will produce a matrix where all the darker elements are close to the diagonal. To overcome this problem, Ling [LING73] combined the display technique with a hierarchical clustering scheme. A display of the matrix, rearranged in the order dictated by the results of the clustering algorithm applied to the original data, quickly reveals whether the clustering was able to produce distinctive classes within the data. While this approach may be somewhat limited, the graphical technique nonetheless serves as an aid to the clustering process and as an alternative to the use of trees or dendograms for displaying clusters.

3.2 Nonmetric Multidimensional Scaling

The objective of nonmetric MDS is to transform a matrix of similarities into a set of data points in a two-dimensional space such that a monotonic relationship between the ranking of the similarities and the distances between points in the metric space is maintained. Shepard [SHEP62a, SHEP62b] and Kruskal [KRUS64a, KRUS64b] developed and refined these techniques in their efforts to draw conclusions about psychological experiments involving judgments as to similarity of stimuli, human behavior, etc. Consider the example of Fig. 3. [GREE70] The data matrix was generated by asking a respondent to judge the similarity of each possible pair of 11 items and then to order the resulting 55 similarity pairs in increasing order of similarity. A value in the matrix represents the ordination number for the corresponding pair of items, with a value of 1 representing the lowest similarity. The resulting MDS transformation of the matrix produced the geometric configuration shown. As may be noted, the graph of the geometric space clearly

	1	2	3	4	5	6	7	8	9	10	11
1	-	48	6	25	44	8	20	54	51	17	48
2		-	18	47	23	19	34	50	52	42	24
3			-	45	1	55	33	10	15	39	4
4				-	12	43	40	37	31	38	14
5					-	2	3	26	28	11	49
6						-	30	9	16	32	5
7							-	27	21	22	7
8								-	53	29	41
9									-	36	35
10										-	13
11											

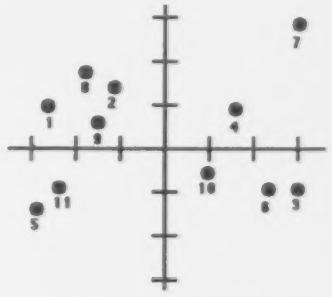


Fig. 3 Example of Nonmetric MDS Mapping

shows the relationship between the items (i.e., the underlying structure) present within the matrix: for example, item 7 is far removed from all other points and items 3 and 6 (with similarity of 55) are very close. The display also reveals the presence of several clusters in the data.

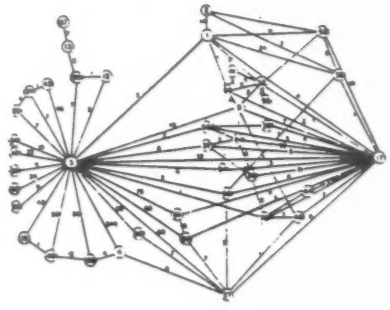
Nonmetric scaling algorithms are conceptually simple, and all are based on the same general procedure. An initial geometric configuration **X** consisting of n points is chosen as the representation of the n vectors whose pairwise similarities have been ranked in increasing or decreasing order. The distance between each pair of points in **X** is calculated, and an assessment is made of the quality of the configuration using some variant of Kruskal's *stress measure*. [KRUS64a] This measure provides an indication of the degree to which the distances between points in the configuration satisfies the monotonicity property defined by the ordering of the similarities. If the monotonic fit is poor, the algorithm moves the points in such a manner that the resulting ranking of distances more closely matches the monotonicity ranking of the similarity values. The stress measure is recalculated for the new configuration, and the mapping process continued until a reasonably good fit (a very small stress value) is obtained. Each iteration produces successively better point configurations.

While mapping techniques may differ in regard to the stress measure or the means by which successive configurations are modified to minimize the stress function, Shepard [SHEP72] notes that most techniques when applied to the same set of data yield virtually indistinguishable results. In fact, Green [GREE70] has shown that as the number of data points increase, the solution for a given dimensionality will become unique.

Small and Griffith [SMAL74, GRIF74] were among the first to reveal structure among documents based on MDS mapping techniques. Their specific objective was to define the structure of science by analyzing *maps* (the document space) of the scientific literature created through the use of cocitation clustering. The underlying principle of their work is that "the stratification systems of science rest, in large measure, on publication activity in professional journals." [DORE85] In this approach, the distance between points in the two-dimensional space is inversely proportional to the strength of the cocitation links between them. The cocitation strength between a pair of documents is the number of other documents within the collection being mapped that cite the given pair of documents in common. Author cocitation analysis is a valuable tool which has been used by several other researchers as a means for revealing the changing structure of scientific specialties. [GARF79, WHIT81, McCA84] For this reason, Small and Griffith's particular mapping process is briefly described.

The original data base used by Small and Griffith consisted of 93,800 natural science articles chosen from 2,400 journals published in 1972. Within this collection, 1,832 documents were subsequently classified as *highly cited* documents (each was cited by at least ten documents in the data base). These documents and the documents citing them, a total of 16,972 documents, were chosen as the collection to be used in the cocitation analysis process.

The collection was initially clustered using a single link clustering algorithm based on cocitation strengths; cluster membership was controlled by the setting of a threshold for the cocitation strength that qualified a pair of documents for inclusion in a cluster. Fig. 4 contains a simple network representation of the resulting clustered space for all clusters containing at least three documents. [GRIF74] In this graph, the labels on the branches represent cluster cocitation strengths, that is, the number of times that documents in corresponding clusters have been cited together. Although the graph does not reflect the size of each cluster, the largest number of papers were in physics (clusters 1 and 2 in the network with a total of 73 articles), biomedicine (cluster 3 of size 801), and chemistry (cluster 17 with 92 articles).



The circles represent clusters containing at least three documents. Numbers within circles were assigned sequentially as the clusters were created.

Fig. 4 Network Diagram of Document Space

To investigate each of these clusters in more detail, a nonmetric MDS mapping was applied to the subset of documents comprising a single *macrocluster*, such as biomedicine. A document similarity matrix was formed, using the number of co-citations of each pair of documents as the similarity measure. The documents were then mapped to a two-dimensional representation using the monotonicity relationship defined by the similarities. An example

of a portion of the resulting spatial display covering methods papers is contained in Fig. 5. The seven clusters indicated in this graph were generated by a clustering algorithm applied to the two-dimensional data produced by the mapping process. An examination of the papers contained within each cluster revealed a logical internal ordering of biomedical methods. [GRIF74]

This approach illustrates how MDS mapping techniques can be applied to larger collections. The combined use of clustering and nonmetric multidimensional scaling of macroclusters provides a means for generating two dimensional representations of the document space.

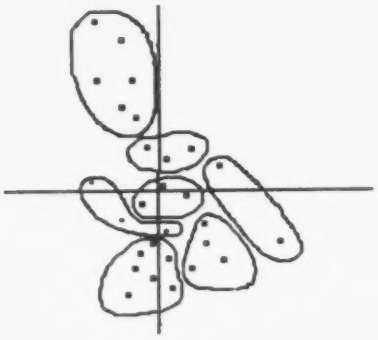


Fig. 5 Mapping of a Clustered Space

3.3 Metric Multidimensional Scaling

The ultimate objective of metric MDS mapping is the same as that of nonmetric mapping, that is, to generate a two-dimensional geometric representation of the original n data items. However, the metric approach requires the availability of the n multidimensional data vectors themselves as defined in the original space, whereas the nonmetric mapping requires only the ranking of the interpoint similarities. The metric mapping attempts to generate a configuration which preserves the structure of the original space by arranging the n points in the lower-dimensional space in a way that approximately preserves their interpoint distances. Thus, each point in the final configuration bears a relationship to all other points similar to that which existed in the original space.

One of the earliest applications of this type of mapping process to a document collection in which each document is represented as a concept vector is described by Sammon in [SAMM69]. The results of his experiment lends support to the validity of the metric MDS technique as a means of generating a graphical representation of the document space. Another noteworthy aspect of Sammon's work is that it is the first report of the actual use of an interactive graphics terminal for displaying a mapping of a document space: the user was able to specify, by means of a light pen, which documents contained in the mapping were considered relevant to the query. The query vector itself was also indicated as a point in the two-dimensional geometric document space. The system then generated a new query based on the user's evaluation. The concept is that "the query vector can be moved to highly relevant regions of the document space by interacting ... with a geometric representation of the space" as has been illustrated in Fig. 2. [SAMM69]

3.4 MDS - General Conclusions

Mapping or ordination techniques produce a geometric representation of multidimensional data. Such graphical displays could prove particularly beneficial by making it possible to locate clusters or outliers present within a set of data. By studying a two-dimensional display of points, one may quickly observe the nearness or remoteness of individual data points within the metric space.

The dimensionality of the data cannot be reduced without the loss of some structure inherent in the data. Thus, while two dimensions may not be generally sufficient to accommodate the full complexion of the relations in a given set of data, applications of multidimensional scaling techniques to document collections have yielded meaningful representations using only two spatial dimensions. Therefore, serious consideration should be given to the use of this type of technique to produce mappings of both the clustered document space and the documents present within individual clusters. Such displays could be particularly useful during an interactive exploratory analysis of a collection.

Of course, the generation of displays such as those in Figs. 3-5 is time consuming. Therefore, transformation methods must be considered which are more efficient than the mapping techniques in the generation of the data point representations.

4. Nongeometric Multidimensional Point Displays

Displays produced by MDS mappings do not reveal the characteristics of the data items which cause the points to be close to or far removed from other points. To obtain this information, one normally must analyze the data at a more detailed level. However, alternate display techniques exist which not only permit conclusions to be drawn about the relations of the data items in terms of nearness or dissimilarity, but also, in some cases, make it possible to ascertain the individual data vector components which

contribute to the observed spacial relations. These methods do not generate a geometrical representation of the data vectors and therefore do not require any computational analysis of the vectors themselves or an iterative refinement of spacial configurations. Thus, they are computationally more efficient than MDS mappings.

The general objective of nongeometric multivariate point graphs is to represent multidimensional data in all its dimensionality. Each of the original n data vectors is translated into a curve or iconic representation in which each dimension of the vector space corresponds to a feature of the icon. The resulting n *pictures* are then displayed for visual analysis. It is assumed that the similarity or dissimilarity of the pictures is directly related to distances between corresponding points in the original multidimensional space. While the final visual display of the nongeometric graphical methods may be more complex than two-dimensional maps of n points, thereby necessitating more user analysis to determine the structure present within the data, other benefits may be derived from these approaches.

The use of these techniques for document space representation has not been reported in the literature. However, we will review briefly such methods, since they are relevant to information retrieval application.

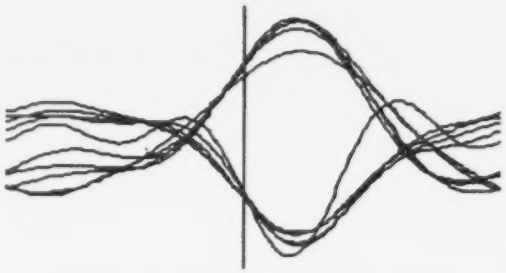
4.1 Andrews' Method

Andrews [ANDR72] proposed a method of plotting multidimensional data in which each data point x is mapped into a trigonometric function of the following form:

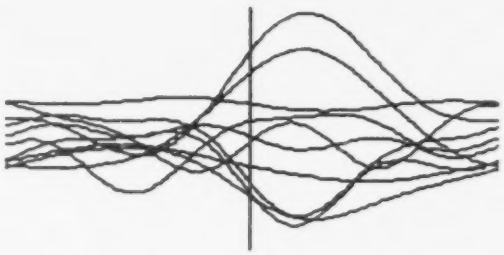
$$f_x(\theta) = x_1/\sqrt{2} + x_2 \cdot \sin(\theta) + x_3 \cdot \cos(\theta) + x_4 \cdot \sin(2\theta) + x_5 \cdot \cos(2\theta) + \cdots .$$

The function is plotted over the range $(-\pi, \pi)$ and is computationally quite simple. Furthermore, it preserves means and Euclidean distances. Thus, points close together in the original space will generally have curves which are close together. This feature enables the display to be used for the possible identification of clusters or outliers present in the data. Fig. 6(a) contains an example of Andrews' method applied to a data base of ten vectors containing two clusters. Andrews' plots of a less homogeneous set of data is shown in Fig. 6(b).

This approach assumes that the more important components of the vector need to be associated with cyclic components in the function which have a low frequency, since in the resulting plots low frequencies are more readily apparent than high frequencies. Thus, proper ordering of the data vector components is important. Andrews suggests that a principal component analysis be performed on the data vectors to determine an appropriate ordering of the data components. Additionally, since a display of a large number of points may be very difficult to analyze, it has been suggested that subsets of the data be displayed to obtain a *feel* for the data.



(a) A Set of Ten Data Items with Two Known Clusters



(b) A Set of Ten Inhomogenious Data Items

Fig. 6 Examples of Andrews' Method

4.2 Iconic Representations

Whereas data points appear as sets of curves in Andrews' mapping technique, in the iconic approach each data point is represented by an icon or symbol whose features correspond to components of the vector itself. Variation in each of the coordinates of the data is represented by a variation in one characteristic of some feature of the icon. A data collection is thus represented by a set of such icons; the display may be examined to reveal similarities, to identify outliers or clusters, etc. This technique has been successfully applied to such diverse areas as offshore oil group performance, mineral exploration, Supreme Court decision trends, psychiatric screening, and urban regional indicators. [WANG78]

Two common iconic representations are *castles* and Chernoff *faces*. In the castles display the height of each parapet is a function of the component it represents. [KLEI81] To maintain a common structure among castles associated with a given data set, the icons are generated by a hierarchical clustering algorithm which assigns dimensions of high covariation to adjacent parapets. In Chernoff faces, each data point is represented by a cartoon of a face whose features, such as length of nose and curvature of the mouth, correspond to components of the data points. [CHER73]

An experimental evaluation of the use of these types of iconic techniques as an aid in making pairwise similarity judgments between items revealed that facial icons were better than all others in revealing the relationships among the data vectors in terms of distances between points. [WILK82] Fig. 7 contains an example of a facial icon which supports the representation of 20-dimensional data; the features of the face are strongly interrelated in order to insure a proper positioning of the features, thereby giving it a more humanlike appearance. [BRUC78]

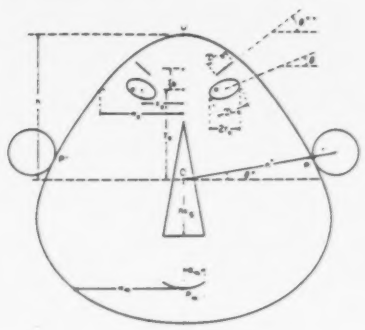


Fig. 7 An Example of the Chernoff Facial Icon

4.3 Component Scale Drawings

Iconic representations while computationally fast are inherently difficult to use in making definitive, numerical scale judgments about the relative distance between the data points generating the icons. This problem does not exist when one uses the MDS mapping procedures or Andrews' method, because both methods produce displays which generally include labelled vertical and horizontal axes. Yet these methods are either computationally expensive or produce complex displays for large data sets.

Component scale drawings overcome these problems to some degree for data sets whose vectors are relatively short (fewer than 100 components) or which can be shortened during the analysis process without sufficient loss of information. [CROU85] In reality, the length of a vector is only limited by the constraints of the graphics display equipment; thus, theoretically, vectors of any length may be displayed in the manner to be described.

Consider the display contained in Fig. 8. This figure shows a query vector (heavy line at the top) and the twenty document term vectors within the document collection which correlate most highly with the query vector. The query vector consists of ten weighted terms with weights in the range (0,1); the weight signifies the importance attached to the term by the query. The abscissa represents the term (component) index and the ordinate, the term weight. In this example, the query vector is the curve with the highest peaks. This display is difficult to analyze. However, as is illustrated by Fig. 9, the display can be modified to reveal the relationships present within the data.

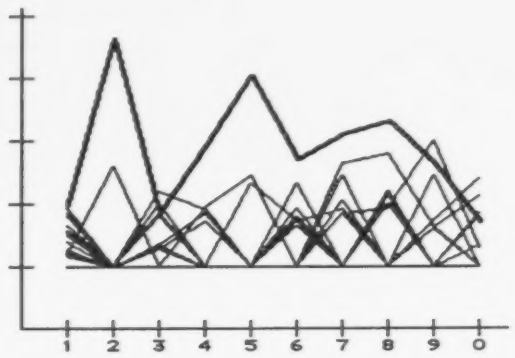


Fig. 8 Component Drawing of the Document Space

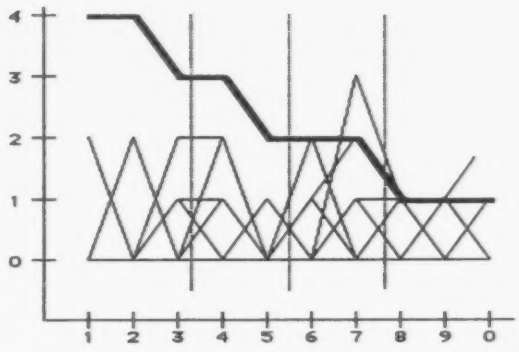
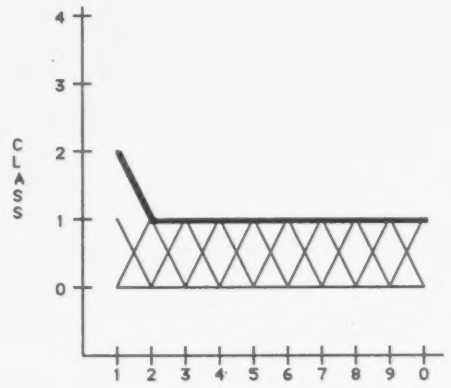


Fig. 9 Component Scale Display of the Document Space

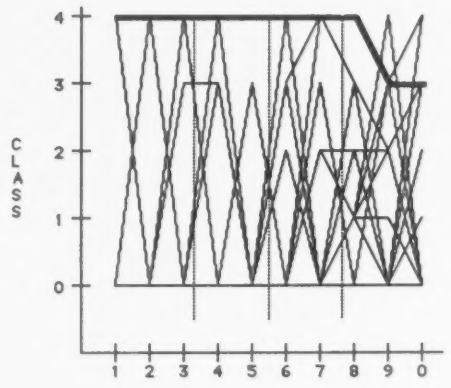
In Fig. 9 the vectors are represented as line drawings with the horizontal axis representing the query terms and the vertical axis, class values indicative of the importance of a term. An abscissa of 1 corresponds to that term in the query vector which has the highest term weight, and an abscissa equal to the length of a query (that is, the number of terms in the query) represents the lowest weighted query term. Thus, the query vector is arranged in a sorted order in decreasing term weight order. An ordinate of any point composing a document may have only one of five possible values (0-4). Each value represents a range of term weights, with class 0 signifying the absence of a query term in the document vector itself (a zero weight). For any class *i*, the weights of all classes with ordinate less than *i* are less than the weights of class *i*, and for any class with ordinate greater than *i*, the weights are greater than those of class *i*. Thus, a term corresponding to a point on the document whose ordinate is class 3 is considered to be more important than a term in that document with a lower ordinate, and less important or indicative of a document's meaning than a term whose ordinate is class 4. It might be observed that this relationship between ordinate classes forces the query to be generally stairstepped downward from the point on the line representing the first query term.

This display is used in an experimental interactive system which permits the user to select and highlight documents or to remove document vectors from the display on command. Depending on the subset of documents being evaluated, the user may desire to see more or less discrimination among the displayed documents. This desire is met by providing a sliding class scale which the user may activate at will; one just clicks on the desired scale and the documents are immediately displayed using an alternate range of term weights for the class ordinates. If one desires to suppress the less significant terms in the vectors, that is, give more importance to the heavily weighted terms, then one can choose a more severe scaling than that which the present display is using. Conversely, by choosing a less restrictive scale, the lower weighted terms are permitted to rise above the base line of ordinate class 1 and to appear as peaks in the display. For example, the user can force the query vector and all documents to be relatively flat at ordinate 1, with no discrimination between vectors (as illustrated in Fig. 10 (a)) or the user can force the query to be a flat line with ordinate class 4, in which case the document vectors will have many peaks (Fig. 10 (b)). Through this dynamic control, one can easily determine the relationship between vectors.

To ascertain the power of this approach in making similarity judgments visually, a series of experiments were performed in which a query vector was displayed together with the twenty vectors which were determined analytically to be most similar to the query. The user was then asked to rank the 20 documents in terms of relative similarity with the query based only on the visual display of the vectors in the form indicated in Fig. 9. This experiment represents a stringent test of the discriminatory power of the display since all the documents are highly similar with the query. Fifteen queries were analyzed; the average Spearman rank correlation coefficient for the 15 cases was 0.85. These results are encouraging and lend support to the use of this display technique. The results indicate that the lack of knowledge of actual weights does not adversely affect the perceived similarity of vectors.



(a) Severe Scale



(b) Less Restrictive Scale

Fig. 10 Effect of Scaling on the Display of Data Items

4.4 Nongeometric Representations - Conclusions

Nongeometric representations can be used to generate displays very rapidly, providing a cursory review of the relations present within a set of data vectors. These techniques appear promising as a means of performing a graphical analysis of the document space. The component scale drawing possesses many of the advantages of both the geometric and nongeometric approaches; that is, it is computationally fast, it provides a quantitative scale by which relative distance judgments can be made, and the display does not become too complex when a large number of data points are displayed.

5. Summary

Information retrieval systems are not currently taking advantage of the immense power of graphical techniques which are available for revealing the relationships existing among documents and queries in the document space. Techniques were examined which appear to have some application in this area. Work needs to be directed toward the use of graphics in iterative search operations.

References

[ANDR72] D. F. Andrews, Plots of High-Dimensional Data. *Biometrics*, **28:3**, 1972, p. 125-136.

[APPL81] S. L. Applegate, *The Use of Interactive Raster Graphics in the Display and Manipulation of Multidimensional Data*. Ph.D. thesis, Department of Mechanical Engineering, Purdue University, Lafayette, Indiana, 1981.

[BARN81] V. Barnett (ed.), *Interpreting Multivariate Data*. John Wiley & Sons, New York, 1981.

[BARN84] V. Barnett and T. Lewis, *Outliers in Statistical Data, 2nd Edition*. John Wiley & Sons, New York, 1984.

[BRUC78] L. A. Bruckner, On Chernoff Faces. In *Graphical Representation of Multivariate Data*, P. Wang (ed.), Academic Press, New York, 1978, p. 93-121.

[CHER73] H. Chernoff, The Use of Faces to Represent Points in k-Dimensional Space Graphically. *Journal of the American Statistical Association*, **68:6**, 1973, p. 361-368.

[CROU85] D. B. Crouch, Graphical Representation and Retrieval of Information - Part I: Visual Assessment of the Similarity of Query and Documents. Technical Report, Department of Computer Science, Cornell University, Ithaca, NY, 1985.

[DORE85] P. Doreian and T. J. Fararo, Structural Equivalence in a Journal Network. *Journal of the American Society for Information Science*, **36:1**, 1985, p. 28-37.

[EVER78] B. Everitt (ed.), *Graphical Techniques for Multivariate Data*. North-Holland Publishing Company, New York, 1978.

[GARF79] E. Garfield, *Citation Indexing: Its Theory and Application in Science, Technology, and Humanities*. John Wiley & Sons, New York, 1979.

[GREE70] P. E. Green and F. J. Carmone, *Multidimensional Scaling and Related Techniques in Marketing Analysis*. Allyn and Bacon, Boston, 1970.

[GRIF74] B. C. Griffith, H. G. Small, J. A. Stonehill and S. Dey., The Structure of Scientific Literatures II: Toward a Macro- and Microstructure for Science. *Social Studies*, **4**, 1974, p. 339-365.

[HEFF85] H. Heffernan, Computer Graphics Technology Outpaces User Skills. *Government Computer News*, **4:9**, May 24, 1985, p. 6.

[HEIN83] K. W. Heiner, R. S. Sacher, and J. W. Wilkinson (eds.), *Computer Science and Statistics: Proceedings of the 14th Symposium on the Interface*. Springer-Verlag, New York, 1983.

[KLEI81] B. Kleiner and J. A. Hartigan, Representing Points in Many Dimensions by Trees and Castles. *Journal of the American Statistical Association*, **76**, 1981, p. 260-269.

[KRUS64a] J. B. Kruskal, Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis. *Psychometrika*, **29:1**, 1964, p. 1-27.

[KRUS64b] J. B. Kruskal, Nonmetric Multidimensional Scaling: A Numerical Method. *Psychometrika*, **29:2**, 1964, p. 115-129.

[LIND85] T. E. Lindquist, Assessing the Useability of Human-Computer Interfaces. *IEEE Software*, **2:1**, 1985, p. 74-82.

[LING73] R. F. Ling, A Computer Generated Aid for Cluster Analysis. *Communications of the ACM*, **16:6**, 1973, p. 355-361.

[McCA84] K. W. McCain, Longitudinal Author Cocitation Mapping: The Changing Structure of Macroeconomics. *Journal of the American Society for Information Science*, **35:6**, 1984, p. 351-359.

[MYER81] R. H. Myers, *Methods for Presentation and Display of Multivariate Data*. Technical Report CR-165788, NASA Langley Research Center, Hampton, Virginia, 1981.

[NICK84] R. S. Nickerson (ed.), *Research Needs on the Interaction Between Information Systems and Their Users: Report of a Workshop*. National Research Council, Washington, D. C., 1984.

[SALT68] G. Salton, *Automatic Information Organization Retrieval*. McGraw-Hill Book Company, New York, 1968.

[SALT78] G. Salton and A. Wong, Generation and Search of Clustered Files. *ACM Transactions on Database Systems*, **3:4**, 1978, p. 321-346.

[SALT83] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, New York, 1983.

[SAMM69] J. W. Sammon, A Nonlinear Mapping for Data Structure Analysis. *IEEE Transactions on Computers*, **18:5**, 1969, p. 401-409.

[SHEP62a] R. N. Shepard, The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function. I. *Psychometrika*, **27:2**, 1962, p. 125-140.

[SHEP62b] R. N. Shepard, The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function. II. *Psychometrika*, 27:3, 1962, p. 219-246.

[SHEP72] R. Shepard, A. Romney, and S. Nerlove. *Multidimensional Scaling*, Seminar Press, New York, 1972.

[SMAL74] H. Small and B. C. Griffith, The Structure of Scientific Literatures I: Identifying and Graphing Specialties. *Social Studies*, 4, 1974, p. 17-40.

[TUFT83] E. R. Tufte, *The Visual Display of Quantitative Information*. Graphics Press, Chesire, Connecticut, 1983.

[VANR79] C. J. Van Rijsbergen, *Information Retrieval, Second Edition*. Buttersworth, London, 1979.

[WANG78a] P. Wang (ed.), *Graphical Representation of Multivariate Data*. Academic Press, New York, 1978.

[WANG78b] P. Wang and W. Lake, Application of Graphical Multivariate Techniques in Policy Sciences. In *Graphical Representation of Multivariate Data*, P. Wang (ed.), Academic Press, New York, 1978, p. 13-58.

[WHIT81] H. D. White and B. C. Griffith, Author Cocitation: A Literature Measure of Intellectual Structure. *Journal of the American Society for Information Science*, 32, 1981, p. 163-171.

[WILK82] L. Wilkinson, An Experimental Evaluation of Multivariate Graphical Point Representations. *Proceedings of the Conference on Human Factors in Computing Systems*, Association for Computing Machinery, New York, 1982, p. 202-209.

IMPROVED SUBJECT ACCESS, BROWSING and SCANNING MECHANISMS in MODERN ONLINE IR

BY

Peter Ingwersen and Irene Wormell

Royal School of Librarianship
Copenhagen
Denmark

ABSTRACT

Focusing on communication, the paper analyses and proposes practical solutions to key problems in online IR, in particular concerned with ill-defined and "muddled" information requirements, concept interpretation in searching and text representation.
The need for development of new additional browsing and scanning feedback devices, based on existing methods to support searchers is emphasised. The paper points to economically feasible indexing methods fitting the potentials of current information technology and adaptable to in-house information environments: the SAP (Subject Access Project) principles. Focusing display mechanisms and the term frequency analysis feature Zoom are discussed and suggested combined into a flexible database front-end. The design principles are outlined and demonstrated in a worked example.

1.0 INTRODUCTION

Modern information technology has opened up several new possibilities of conceptual design of IR systems that more fully reflect actual searching behaviour. Thus, future indexing and retrieval procedures should be revised and based on a better understanding of human communication.

The main objective of information retrieval is to facilitate the effective communication of desired information between human generator and human user.
This objective necessitates an interest among information scientists and IR systems designers for the following 5 areas (BELK78):

1. information in human, cognitive communication systems;
2. the idea of desired information;
3. the effectiveness of information (systems) and the effectiveness of information transfer;
4. the relationship between information and generators;
5. the relationship between information and user.

This view of IR emphasises the individual, cognitive nature of information inquirers, intermediaries, and the members of the family of information generators, authors etc. (INGW84).
In addition, the view points to problems of concept interpretation in each single step in the IR process, and between the individuals involved in IR.
Today's online IR systems are designed and operate within the framework of the "best match" principle. As discussed by (BATE85) and (INGW86) among others, this principle has been a valuable experimental model. It has taught information science much and is helpful to a significant portion of searches where the need can be articulated in a straightforward way. In such situations, relevance feedback and search formulation modifications are all geared to that final point in which query and document representation are matched to produce the retrieved set. But the matching principle is definitely inadequate to describe all of the IR situations that manual and online systems should be able to accomodate.
Focusing on communication, the paper aims to analyse and propose practical solutions to key problems in online IR, in particular concerned with ill-defined and "muddled" information requirements, concept interpretation in searching and text representation.
The need for the development of new additional browsing and scanning feedback devices, based on existing methods to support the user from the query definition throughout the search process is emphasised. In particular, the term frequency principles of the "ZOOM" feature are stressed. Browsing is understood as undirected, often random but active searching for the definition of an information requirement. With scanning is meant directed, active searching for alternative or more refined search concepts. The paper points to economically feasible and communication-based indexing methods fitting the potentials of current information environments as well as commercial online vendor systems.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

2.0 Analysis of key problems in IR

Stripped to its essentials, 'best match' is based on the following fundamental assumptions:

a) that cognitive structures in IR are of paradigmatic, collective nature - not seen as individually different knowledge structures;
b) that the individual user is able to specify the required information;
c) that the information need expressions are functionally equivalent to document texts, i.e. equivalent to information or knowledge;

Collective cognitive structures, often described in paradigm theories of behavioural science, most explicitly by (KUHN70), may influence both the structure of indexing systems and the conceptual patterns of generators and searchers. Thus, IR is only likely to be effective when the individuals involved share common knowledge structures (KS). Both assumptions b) and c) are basically erroneous. The ability to specify requirements may be correct if an intermediary mechanism participates, obtaining knowledge via interviewing about the problem behind the initial request.
Investigations (BROO84, INGW82) indicate that the initial request often exists in the form of a label, consisting of few concepts more generic than or out of context of the underlying requirements (the Label Effect).
Assumption c) is closely related to the fundamental problem regarding IR systems: concept interpretation and feedback in terms of contextualisation.
In traditional online IR and particularly in in-house management and retrival of information, three basic types of information requirements exists, depending on the cognitive conditions of the searcher in the actual problem solving situation.

1. Verificative or locational information problems, i.e. the user wants to verify or locate items - e.g. some specific reports or GMPs. Bibliographic data e.g. ID no., source, pages, paragraphs, author or title - are in this case known to the user.
2. Conscious topical information problems, i.e. the user wants to clarify, review or pursue a subject area including retrieval of facts within a known subject matter. The conceptual KS contain the concepts and terms necessary to express the required potential information.
3. Muddled topical information problems, i.e. the user wants to explore some new concepts or concept relations outside known subject matter. For instance, a scientist wants to trace analogue theories and possible solutions applied in other disciplines and he has no real idea of how to express the problem. The conceptual KS are weak (or muddled) or non-existent with respect to the required potential information.

In theory, the Label Effect may appear in type 3, rarely in type 2 and never in type 1. In practice however, the effect is common in both types 2 and 3 cases - as demonstrated by (BROO84) and (INGW82).

2.1 Proposals for improved IR

We propose to support the scattered types of IR customers on two fundamental steps in the retrieval process by:

- helping the searcher to define the information required, essentially by browsing contexts based on feedback of natural language representations:
- aiding the searcher in concept interpretation of system output by introducing feedback mechanisms based on natural language and controlled vocabulary representations.

By combining existing indexing methods with online feedback and output features currently in action, one may enable an IR system to supply the possibilities needed for searchers to discover unanticipated concept relations and to explore search pathways, not imagined beforehand.

First of all, a front-end device is required containing a filter to sort out which type of information problem that shall be met. Secondly, type 1 problems can be dealt with by menu-driven search processes (as a supplement to traditional catalogue searching). Further details on the filter device and the role of automated interactive intermediaries are recently explored by (VICK84) and (INGW86).
Preferably, type 2 searchers ought to be guided back to the underlying problem by intermediary mechanisms or persons, in case of appearance of Label Effects. With no such intermediary present, the searcher must be guided in his definition process by conceptual maps or contexts, mainly consisting of natural language representations. His browsing time will probably be short, followed by really effective search processes. During the latter he should be able to switch between natural language representations and preferred system terms by active scanning of conceptual feedback and system output.
Type 3 searchers require browsing facilities, enabling random selection of concepts. "Muddled" types of needs require facilities that make it possible (i) to pin-point concepts the searcher vaguely knows are important (he knows his KS are incomplete), and (ii) to discover new, hitherto unknown concepts or search directions (leading to serendipity effects). The facilities proposed are contextual strings of natural language representations with links to thesaurus terms and structures as well as to frequency ranked lists of natural language and controlled terms.

The indexing principles are based on the SAP approach (Subject Access Project), experimentally developed by (ATHE77,78) and fully advanced and put into commercial use by Wormell in her doctoral thesis on Improved Subject Retrieval for Monographic Publications (WORM85). Further details are given in sections 3.1 and 3.2.
Feedback and output mechanisms are currently in commercial operation. For example, the term frequency analysis facility Zoom on ESA-IRS which displays ranked lists of terms applied to selected records (INGW84). Other feedback features are the Focusing Device on I-RIP and BRS-SEARCH, allowing for display of exhibited search terms in context. Salton's relevance feedback approach, in which

the inquirer's reaction to initial retrieved data is used to modify the query until the optimal formulation is found (SALT83), is a more subtle solution and may improve the IR situation, but does not change it dramatically. These ways of treating already entered search concepts lean to 'best match' retrieval by presupposing that information requirements are always specifiable. By acting as stand-alone facilities in different host systems the potentials in modern information technology are not fully exploited.

Hence, we propose to combine the potentials of the contextual Focusing feature and the term frequency analysis facility ZOOM, see the sections 3.2 and 3.3, with SAP indexing principles into an integrative approach to IR system design.

3.0 FORMS of REPRESENTATIONS

As indicated, concept interpretation problems may occur when we try to represent cognitive structures generated by authors and when presenting searchers with the results. Both in form of input to and output from IR systems one must take into account the receiving mind - whether computerised or human.
Putting bibliographic work into a widened context, indexing and the different methods for document representation are key instruments in knowledge communication and in the social utilization of document records. Today's documentary IR systems, however, are lacking consciousness of the functions of these routines as basic communication processes in the transfer of knowledge, mediating information from author to reader.
In spite of the fact that the application of computers in IR has opened up quite new possibilities for internal and external representations in information systems, the representation used in many computerized systems have simply been adopted from those used in printed tools - without an analysis of their suitability for machine-readable files. Since the justification of any indexing technique ultimately lies in the retrieval result obtained, the type of vocabulary used for the representation must be decided from the practical points of the system's use; that is, for which kind of information problems and potential users it should be used.

Traditionally, most bibliographic databases are believed to be used for providing physical and bibliografic access to the recorded knowledge of mankind, normally, with the need for trained intermediaries implied (WORM85). Because of the radical changes in today's information landscape, the potential use and user(s) of IR systems are placing quite different demands (also) on representation forms and the intellectual accessibility of document collections. The retrieval performance aims to supply the searcher with the various types of representations which are the results of subject analysis carried out at different levels and with different methods.

For solving the information problem type 2, described above in section 2.0, classification codes, keywords, descriptors and other types of controlled vocabularies are useful instruments to retrieve information in a known subject area.

The searcher, however, who is navigating in a less known area, probably benefits more from a directory or map which tells him what there is to know, the various viewpoints and how to explore them, than receiving answers to questions which were unnaturally specific and only seemingly well-defined. Therefore, modern IR systems are offering also natural language representation forms (resulting from an exhaustive and deep indexing or from full text input).

With the exception of the mentioned focusing and term frequency facilities, most forms of system output have been adopted from those applied in printed tools, similarly without an analysis of their suitability in different IR situations. In particular, to handle the various types of in-house data carriers and information needs require very flexible file organisations and forms of output representations.

3.1 SAP indexing

SAP indexing is aimed to improve the subject description of books by representing in detail the versatile and distinguishable contents of monographic publications. Using the Table of Contents, back-of-the-book-index, and other "inherent" attributes, one selects terms and phrases which point to the significant subjects discussed. Indexing terms are arranged and processed so, that one locates not only the book, but the specific page(s) within the book on which that subject is discussed (see fig. 1).
This innovative indexing technique has been applied to a specific type of monographic publication, Swedish Government Official Reports to explore the rich resources of factual information in this publication. By indexing Table of Contents, the legend and captions of tables and graphs, and other significant parts of the reports, one may in addit on search for very specific factual and numerical data - often buried in the publication and scattered throughout the text. Rules and guidelines have been devised to ensure that SAP indexing does not require any special subject knowledge on the part of the indexer. All the extracted terms can be retrieved in free text mode.
The main idea of SAP makes advantage of the improved online retrieval capabilities and gets around the cost problem of deep indexing which is primarily one of intellectual effort.

3.2 Search facilities related to SAP indexing

Looking at the user, the types of requirements and his effort to use the system to find information captured in documents, an important criterion for efficient communication is that the terms used for content identification are represented in contextual strings - as opposed to the use of single terms or short phrases, known as keywords, index terms or descriptors.
In the latter case, the relations between terms and the specific aspects of the treated subject are rather vaguely or not at all indicated.

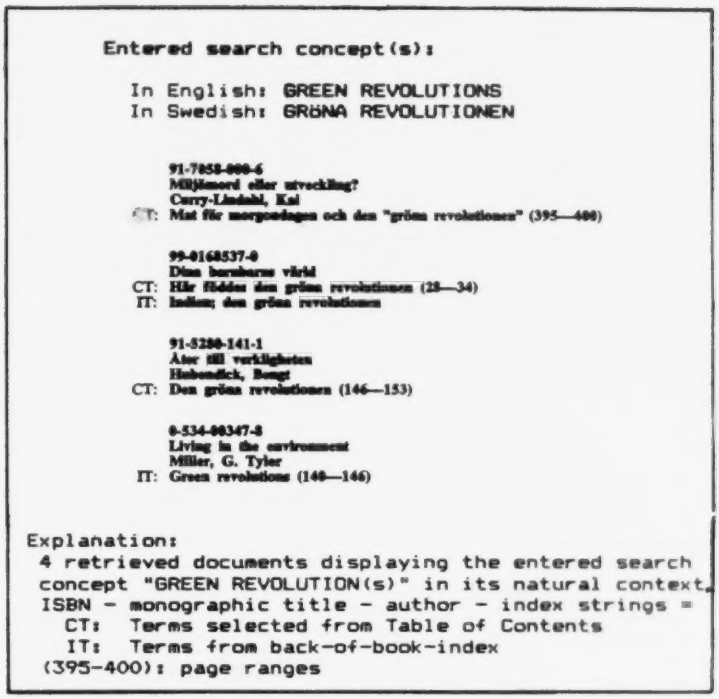

Entered search concept(s):
In English: GREEN REVOLUTIONS
In Swedish: GRÖNA REVOLUTIONEN
91-7858-000-6
Miljömord eller utveckling?
Carry-Lindahl, Kai
CT: Mat för morgondagen och den "gröna revolutionen" (395—400)
99-0168537-0
Dina barnbarns värld
CT: Här fiddes den gröna revolutionen (25—34)
IT: Indien; den gröna revolutionen
91-5280-141-1
Åter till verkligheten
Hubendick, Bengt
CT: Den gröna revolutionen (146—153)
0-534-00347-8
Living in the environment
Miller, G. Tyler
IT: Green revolutions (140—146)
Explanation:
4 retrieved documents displaying the entered search concept "GREEN REVOLUTION(s)" in its natural context.
ISBN - monographic title - author - index strings =
CT: Terms selected from Table of Contents
IT: Terms from back-of-book-index
(395-400): page ranges

Fig. 1. Example of the FOCUSING mechanism of the Swedish SAP database on 3-RIP.

The fact that SAP entries are always consisting of terms in context - representing strings such as chapter headings, captions to tables and graphs... etc. - facilitates the searcher both in browsing situations and in relevance feedback, see Fig.1, displaying focused SAP entries.
However, the contextual representation and the use of exhaustive indexing often leads to long and extensive bibliographic records which put some special demands on the organisation of files and flexible output formats. To facilitate quick browsing and relevance judgement of the retrieved items, a special technique has been developed in the SAP databases: focusing on the retrieved contextual strings and the related bibliographic data presented in a short form, see Fig. 1.

Being well aware that inexpensive automatic text processing often leads to the accumulation of enormous text masses placing great burdens on both the system and its users, it must be pointed out that:

- SAP-indexing, similar to other natural language applications in IR, must be supported by efficient retrieval tools which accomodate the user in all the different phases of the search procedure;
- linguistic and other studies of cognitive processes are required for the (future) replacement of long texts by denser representations; we have to avoid inflation of mass text processing, to reduce the databases quantitatively and improve them qualitatively.

The Zoom term frequency facility described in section 3.3 is an example of density, already in commercial use.

In spite of some present shortcomings of SAP-indexing, it is worth noting as a useful "hint" for catalogue developers in many different environments. Since this straightforward and simple method of capturing the presence of data in publications does not call for the subject specialist's analytical work in deep indexing, it is suitable for building local databases at modest costs.
It is also a useful representation form for subject contents in heterogeneous user environments and document type landscapes in which the use of mets-languages for document retrieval normally render great difficulties for the end-users.

3.3 The term frequency analysis feature Zoom

Zoom forms part of the ESA-IRS command language system ESA-QUEST. Zoom was invented in order to improve and support the casual user's search performance (MART82). The feature is designed to analyse the frequency of single words, phrases or

codes appearing in a selected set of references. In the basic version a maximum of 200 records can be analysed. It is based on the converted versions of the large-scale IRS databases. For each the following searchable fields are made analysable:

titles, abstracts, authors, journal & institutional names, classification codes, controlled and uncontrolled terms (=index terms). In certain files additional special fields are zoomable, e.g. molecular formulae and registry numbers on Chemabs.

The frequency analysis is by default carried out on the index terms in fifty records across the last selected set. In its simplest form the command ZOOM or Z thus implies frequency analysis of all index terms - phrases or single keywords - applied to fifty records across the last set. The outcome is a ranked list of concept designations associated with an entered search statement, and in one way or another related to each other. See Fig. 2.
Other command functions exist, for instance: ZOOM LATEST - i.e. analysis of index terms of the 50 latest refs. in last set; Z TITLE WORDS - analysis of single document title words. More elaborated command functions are described and discussed by (INGW84). Depending on the form of the search statement, the command ZOOM TITLE may result in a display of entire document titles incorporating the search terms, very similar to the SAP focusing shown in Fig. 1. No additional data are, however, displayed. See Fig. 5 & 7. The interesting aspect of ZOOM is the fact that the frequency analysis feeds back a conceptual cognitive structure representing the actual relations between the entered search statement and the ranked terms.

For example, a searcher may select "micro computer(s)" or "microcomputer(s)" and "CAD" as title words on Inspec (retrieving 13 refs.) and analyse the records for index terms:

```
Text Analysis Results
Frq Words/Phrases
--- ------------------------------------

 11 CAD
  4 MICROCOMPUTERS
  3 CIRCUIT LAYOUT CAD
  3 COMPUTER AIDED DESIGN
  3 MANUFATURING COMPUTER CONTROL
  3 MICROCOMPUTER
  3 MICROCOMPUTER APPLICATIONS
  2 CIRCUIT ANALYSIS COMPUTING
  2 CIRCUIT CAD
  2 EDUCATION
  2 INTERACTIVE TERMINALS
  2 LOGIC DESIGN
  2 MANUFACTURING DATA PROCESSING
  2 PRINTED CIRCUITS
  2 ROBOTS
  1 ARTWORK DESIGN
  1 CAD CAM
  1 COMPUTERISED CONTROL
...Pages.Lines: More=  2.6
```

Fig. 2. Zoom on index terms, Jan. 1986. Inspec. (2 screen pages & 6 lines still available)

The ranked list, or conceptual KS, shall be interpreted as follows: "COMPUTER AIDED DESIGN" occurs 3 times and "CIRCUIT CAD" twice, each as index terms in the analysed set of references - related to the search term combination in natural language.
Because of the high frequency of the preferred terms, e.g. "CAD" and "Computer Aided Design", it is rather easy even for beginners or persons with scarce subject knowledge to pin-point the adequate terms and add them to the search, hereby improving recall in OR combinations with the original search term.

Consequently, the Zoom feedback makes heuristic searching available, allows for browsing, scanning and recognition, and points to for example, spelling variants (computeri?ed) or more specific conceptual relations within a database, e.g. search entries like "circuit layout CAD" & "circuit CAD". In addition, it may suggest new terminology. It may give ideas and associations in precision of incomplete KS, serendipity effects (e.g. "manufacturing...") and work as relevance feedback tool. As an information research and management tool it is interesting. For instance, Zoom quickly provides clusters of the core journals, institutions or researchers in particular R&D areas.

4.0 OUTLINE of the PROPOSED BROWSING and SCANNING MECHANISMS

By combining the SAP deep indexing method with Focusing and Zooming principles into an integrated and flexible part of a host mainframe or in-house database front-end it is possible to support searchers with different types of information requirements, see Fig. 3.

Depending of the actual knowledge state of the searcher, i.e. possessing a type 2a, type 2b or type 3 information requirement, the two mechanisms A and B may function differently. Type 2 searchers may browse for a very short time (or not at all), and, via scanning of mechanism B displays, obtain alternative or more precise search terms.

Type 3 searchers may fully explore the A mechanism and via different options - see below - for a period of time interact with and orientate themselves in the potential conceptual structures displayed, hereby step by step defining their requirement. This browsing period may be entailed by applications of mechanism B options, more suitable for scanning.

Together with document titles the SAP entry strings provide conceptual contexts in natural language. A number of controlled terms (or at least class code names) added to the documents are consequently linked through the indexed items. Preferably, but not mandatory, an online thesaurus is linked up with the file.
To form the dense semantic structures necessary for the searcher to use for browsing and scanning frequency analysis of terms is applied to the natural language and index terms, eventually leading to a thesaurus. The existing Zoom feature default is changed into ZOOM LATEST 50 records.

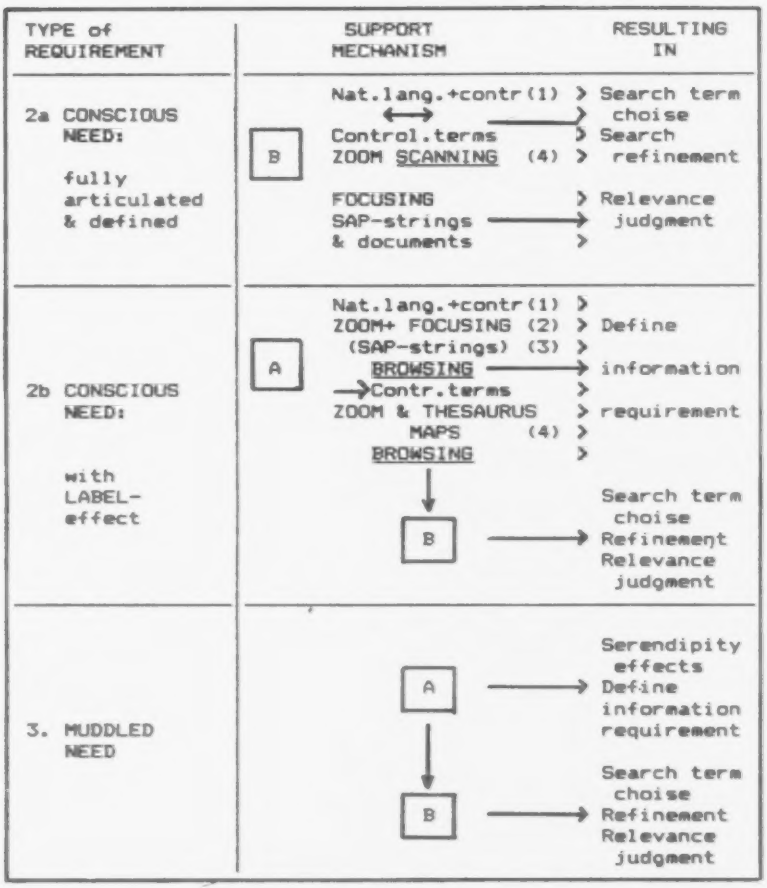


Fig. 3. Mechanisms accomodating definition of information requirements and concept interpretation. (1)-(4) refer to zoom options.

Four options are allowed to exist for zooming, automatically executed depending on where in the browsing/scanning sequence the searcher chooses to be:

1. ZOOM LATEST TITLE WORDS & INDEX TERMS, i.e. an analysis of the 50 latest items' title, SAP string words and index term phrases, which is executed by default when initial search statement is entered. The ranked term list displays natural language words mixed with controlled conceptual phrases, making browsing and exploration possible, and therefore suitable for type 2b and 3 searchers. See Fig. 4.
2. ZOOM LATEST TITLE WORDS, i.e. an analysis of title and SAP string words of the 50 latest records, executed if the searcher chooses a natural language option via screen menu. The option may automatically analyse words related to the initial statement or a new search statement selected from previous displays. Two different lists may be displayed: one with words from strings containing the search statement; alternatively, one listing words from associated strings which are conceptually more remote from the search statement.
3. ZOOM LATEST TITLES, i.e. a dense display of entire item titles and SAP strings, and executed if the searcher wants a contextual, natural language option. By default it ranks the strings containing a combinative search statement of minimum two words to minimise the number of strings. Again, the different lists are available. This analysis is optional after option no. 1 has been executed and may serve as a browsing device. See Fig. 5 or 7.
4. ZOOM LATEST TERMS, i.e. an analysis of the 50 latest items' index words and phrases, executed if the searcher chooses a ranked list of purely controlled term phrases via screen menu. This option may by default analyse co-occurent concepts associated with the initial statement or a new one selected from previous displays, and entered in free-text mode to maximise term variety. It may

serve as both browsing and scanning device. Conceptual links from end-user vocabulary, (perhaps) in natural language, to legitimate database terms are hereby established, see Fig. 2.

Searchers with "conscious" needs with label effects and "muddled" requirements may explore the options 1 to 4 by browsing. Automatically, the software shall store the sequentially selected statements to be displayed when required by the user. The 3 first options form the main part of mechanism A in Fig. 3.
Mechanism B contains option 1 and 4 plus the Focusing device shown in Fig. 1. Mechanism B is available to the "conscious" requirement users, having a defined problem ready. Via option 1 such searchers may scan the ranked lists, recognise concepts which may trigger new related concepts, expand to preferred database concepts by option 4 and focus (or zoom further) for relevance judgment purposes.
(Expert) searchers wishing to skip the front-end may do so and perform traditional IR.

4.1 Worked example

An example of sequential displays in relation to a muddled information requirement is outlined below: (We have not sufficient space to develop the conversational parts of the interface situation nor to propose the use of the filtering device).
A Danish industrial designer wants to get some impressions of possible use of microcomputers. He enters what he believes is the correct English form of "microcomputer": MICRO COMPUTERS; the front-end automatically truncates the statement or uses more sophisticated stemming algorithms in free-text mode to maximise the variety of retrieved (& ranked terms); it then executes option 1, see Fig. 4.
The entered terms are exhibited automatically and keywords are supplied with asterisk (*). Menu pointers to proceeding options are not shown. The designer selects "microcomputer(s)", frequency 17 & 14, since these variants seems relevant. Also, he becomes interested in "design", "architecture" and the term "CAD", the latter reminding him of something a collegue told him, however presently

```
KEYWORDs (*) & single TITLE words associated with:
   >> MICRO COMPUTER(s)                    (=set 1) INSPEC
Frq Words & Phrases:            Frq Words & Phrases
--- ------------------------    --- ------------------------
 79 COMPUTER                     14 MICROCOMPUTERS *
 66 MICRO                        13 CAD *
 53 SYSTEM                       12 CONTROL
 23 COMPUTERS                    11 64
 23 SYSTEMS                      11 MICROCOMPUTER
 18 SOFTWARE *                      APPLICATIONS *
 17 MICROCOMPUTER *               9 GRAPHICS
 16 DESIGN                        5 MICRO COMPUTER *
 15 MINI                          3 COMPUTER ARCHITECTURE*
```

Fig. 4. Option 1 sample (manipulated).

The screen menu options will drive the searcher from one semantic structure to another, including thesaurus maps when available, see e.g. Fig. 6. Finally, the zoomed lists always should inform about which search term(s) or set combinations that are analysed for associated terms, which option that is executed and which database one is currently searching (on host mainframes). At present, this is not the case on IRS.

forgotten.
Hence, he decides to explore "CAD" and chooses to browse through contextual strings in natural language (= option 3).
The front-end automatically enters: FIND CAD/TI - which retrieves all document entity titles and SAP strings containing "CAD". Also, it executes option 3 (he rejects the search term combination default offered):

```
TITLE STRINGS containing:
                    >> CAD                      (=set 3)
Frq Term strings                                 INSPEC
--- ------------------------------------------------
  1 A GRAPHICAL MAN MACHINE INTERFACE FOR CAD AND
    SIMULATION OF DYNAMIC SYSTEMS
  1 A MICROCOMPUTER BASED CAD CAM SYSTEM FOR LOW
    COST AUTOMATION (775-94)
  1 CAD APPLICATIONS FOR SPACECRAFT SYSTEM EMC
  1 CAD CAM SYSTEM DEVELOPMENT
  1 CAD FOR INTEGRATED CIRCUITS
  1 CAD GIVES BECKER MUCH NEEDED LIFT
      :
  1 CAD OF THE PERIPHERY OF PLANAR JUNCTIONS
...Pages.Lines: More= 2.7
```

Fig. 5. Option 3 example (only single search term analysis).

The searcher browse through the strings but can't figure out what "CAD" really is - something both related to tennis and astronomy??
Since the Inspec file makes available an online thesaurus, he enters "CAD" guided by the thesaurus option. The front-end automatically locates if the term is contained in the thesaurus structure, see Fig. 6, displaying the traditional layout slightly explained. Preferably, the layout may visualise structured maps, not indicating hit figures.

He discovers that the full-text synonym is "computer aided design" and then remembers the implications of the concept. In addition, he reviews the term "graphics" (computer graphics) and observes that in fact "architectural CAD" exists as a concept, legitimated by Inspec. The designer now feels he has got hold of his requirements for information and displays his previously selected terms (e.g. "micro computer?"; "microcumputer?"; "CAD"; "computer aided design"; "computer graphics"; "architectual CAD"). From the list he may continue his search, for instance applying option 4 scanning Fig. 2 - followed by option 3 for relevance judgment - see Fig. 7 - and focusing, Fig. 8. Obviously, the searcher has now surpassed the A mechanism and may try to develop his search further.

Inspec Thesaurus structures associated with:
>> CAD (=set 3)
N= narrow term R= related term B= broader term
T= top term S= synonym with first term

Ref	INDEX-TERM	STRUCTURE	ITEMS	R
R1	**CAD**.....		15180	15
R2	ARCHITECTURAL **CAD**.....	N	517	6
R3	CIRCUIT **CAD**.....	N	2682	7
.	.	.	.	.
R9	COMPUTER GRAPHICS.....	R	8682	9
.	.	.	.	.
R13	COMPUTER APPLICATIONS.....	T	4619	15
.	.	.	.	.
R16	COMPUTER AIDED DESIGN.....	S	2595	

Fig. 6. Extract from Inspec Thesaurus (layout modified).

TITLE STRINGS containing:

>> Set 7 (Set combination)

Frq	Title strings	INSPEC
1	A **MICROCOMPUTER** BASED **CAD** CAM SYSTEM FOR LOW COST AUTOMATION (775-94)	
1	MATIC **CAD** COMPUTER AIDED DRAFTING ON A **MICROCOMPUTER** (118-23)	
1	**MICROCOMPUTER** BASED **CAD** PROVIDES LOW COST DESIGN TOOLS	
1	**MICROCOMPUTER** BASED ENGINEERING WORKSTATION FOR **COMPUTER AIDED DESIGN**	
1	**MICROCOMPUTER** GRAPHICS A PATH TO LOW COST **CAD** CAM	
1	MOSAIC A **CAD** FEM SYSTEM ON MINI **MICRO COMPUTERS** (189-200)	

Fig. 7. Sample of option 3, modified extraction.

```
CHAPTER HEADINGS & DOCUMENT TITLE display of:
                    >> Set 7        (set combination)
Display                          Revised cronolog. order
---  ------------------------------------------------
 1   85C017097  Conf
     Engineering Software for Microcomputers,1984
     MOSAIC: a CAD/FEM system on mini-micro
      computers (189-200)
     Knopf-Lenoir, C. et al.

 2   84B058626  Conf
     Advances in Instrumentation. Vol.38, Part 2,
      1983
     A microcomputer-based CAD/CAM system for
      low-cost automation (775-94)
     Chen, J.Y.F. et al.
```

Fig. 8. Focusing sample, manipulated.

Searchers with more "conscious" types of requirements may not need to pass all the demonstrated steps. However, the "label" searcher type (2b, fig. 3) may require a short period of time browsing, e.g. fig. 4 (option 1) followed by fig. 2 (option 4) to produce the concepts defining his information requirement which are stored in the mind, but not articulated. From an observers point of view it may be difficult to distinguish between a "label" searcher type and a "muddled" type in the initial stage of searching. The difference derives from the mental performance when applying the different options available.

5.0 CONCLUDING REMARKS

With these rich and varied display possibilities, in particular searchers with muddled or ill-defined requirements may be stimulated to follow various (unexpected) lines of thought. Options and menu tables are the means offered, pointing to natural language contexts, index term phrases and thesaurus structures - or to mixtures. Certain options involve automatic execution of term frequency analysis. In addition, traditional displays of single records, parts of records via "locating" and common retrieval procedures are available via additional pointers. Preferably, pointers are displayable in form of "windows".

An in-house system will need one dedicated browsing and scanning front-end mechanism, whereas it is desirable for online host mainframes to make available a master front-end allowing for cross-searching of several databases as suggested by (BATE86). Certain concept interpretation problems are currently existing for the various cross-file searching features in action - but can be overcome. The proposed browsing mechanism can eventually lead to more elaborated IR supporting or expert systems which may incorporate learning devices (INGW86). Also, it is necessary to involve linguistic approaches to semantic mapping or stored texts as well as user statements, as developed, for instance, by Canani in relation to full-text legal databases (CANA86).

REFERENCES:

[ATHE77] Atherton, P., "Improved Subject Access to Books in Online Library Catalogs". Proceedings of the 1st International Online Information Meeting. London, 1983. pp. 131-138.

[ATHE78] Atherton, P., "Books are for Use": final report of the Subject Access Project. Syracuse University. Syracuse, N.Y., 1978.

[BATE86] Bates, M.J., "An Exploratory Paradigm for Online Information Retrieval". Proceedings from IRFIS 6 Conference, Frascati, Sept. 1985 on Intelligent Information Systems for the Information Society. To be published by North-Holland, 1986.

[BELK78] Belkin, N., "Information Concepts for Information Science". Journal of Doc., Vol. 34 no. 1, 1978. pp. 55-85.

[BROO84] Brooks, H.M., Daniels, P.J. and Belkin, N., "Using Problem Structures for driving Human-Computer Dialogues". September 17. Private communication. 21 p. + appendices, 1984.

[CANA86] Canani, V.B., "Sintesi Grafiche nello "Information Retrieval"". Proceedings from the IRFIS 6 Conference, Frascati, Sept. 1985 on Intelligent Information Systems for the Information Society. To be published by North-Holland, 1986.

[INGW82] Ingwersen, P., "Search Procedures in the Library Analysed.from the Cognitive Point of View". Journal of Documentation, Vol. 38, no. 3, 1982. pp. 165-191.

[INGW84] Ingwersen, P., "A Cognitive View of three Selected Online Search Facilities". Online Review, Vol. 8, no. 5. 1984. pp. 465-492.

[INGW86] Ingwersen, P., "Cognitive Analysis and the Role of the Intermediary in IR". Intelligent library & information systems: progress & prospects. To be published by Harwood, 1986.

[KUHN70] Kuhn, T.S., "The Structure of Scientific Revolutions". 2. ed. Chicago, University of Chicago Press, 1970.

[MART82] Martin, W.A., "Helping the less Experienced User". Proceedings of the 6th International Online Information Meeting. Oxford Learned Information, 1982. pp. 67-76.

[SALT83] Salton, G. and McGill, J.M., "Introduction to Modern Information Retrieval". New York: McGraw Hill, 1983.

[VICK84] Vickery, A., "An Intelligent Interface for Online Interaction". Journal of Information Sc., 9, 1984. pp. 7-18.

[WORM85] Wormell, I., "Subject Access Project: (SAP): Improved Subject Retrieval for Monographic Publications"(Doc.Thesis) Lund: Lund University, 1985.

S-Tree: A Dynamic Balanced Signature Index for Office Retrieval

Uwe Deppisch

Computer Science Department
Technical University Darmstadt
D-6100 Darmstadt, West-Germany

Abstract - The signature approach is an access method for partial-match retrieval which meets many requirements of an office environment. Signatures are hash coded binary words derived from objects stored in the data base. They serve as a filter for retrieval in order to discard a large number of nonqualifying objects. In an indexed signature method the signatures of objects stored on a single page are used to form a signature for that page. In this paper we describe a new technique of indexed signatures which combines the dynamic balancing of B-trees with the signature approach. The main problem of appropriate splitting is solved in a heuristic way. Operations are described and a simple performance analysis is given. The analysis and some experimental results indicate a considerable performance gain. Moreover, the new S-tree approach supports a clustering on a signature basis. Further remarks on adaptability complete this work.

1. Introduction

1.1 Background

In a modern office environment the retrieval of objects by content is one main requirement the data base system layer of an office information system should support [CHRI84, RABI85]. Generally these objects have a variable length and a complex structure. Besides simple data types like fixed length character strings, real and integer, there appear types of variable length; e.g. formatted data like multivalued attributes and especially unformatted data like text [SCHE81, SCHE82, GIBB83]. Figure 1 gives an example for these mixed type office objects. Here, *name* and *delivery* are atomic valued attributes, whereas the attribute *order* is of the set type and *order text* is of type text.

Single-level signatures are applied to support the retrieval of these office objects [CHRI84, FALO84]. Compared to alternative access methods [cf. SALT83b, RABI84, FALO85a], signatures offer the advantage to process partial-match, subset match, substring match and fussy match queries in an easy manner [cf. HARR71, RIVE76, SCHE77, SCHE78]. Because of their simple maintenance [cf. RABI84], signatures are well suited as an access path in data base systems in the environment of server-workstation networks.

Signatures are hash-coded binary words of fixed length; they represent abstractions of objects. Generally all bits of the signature are cleared to *null*, then a hash transformation is applied to the object's values to determine which bits are *set* to *one*. Several proposals have been made for the coding of signatures especially for textual attributes [e.g. HARR71, ROBE79, FALO85b, FALO85c], but their details are beyond the scope of this paper. Here, more emphasis is laid on the physical organisation of the signatures. In single-level signature methods the signatures are stored separately from the objects and searched sequentially for retrieval purpose. Therefore, the same coding transformation is applied to the query to get the query signature. A scan of the signature file returns those object identifiers whose signatures contain *ones* in the positions the query signature does. After these match candidates are fetched from secondary storage a comparison with the query specifications finally eliminates false drops.

1.2 Organisation of Signatures

In single-level signature methods every signature must be accessed and tested. This is done faster than sequential scan of the objects themselves, because the signatures are much smaller. Obviously the retrieval performance of the first step is linearly related with the number of objects. The more objects exist the more time is spent on scanning the signature file [cf. RABI84].

This research was supported by the Scientific Center Heidelberg of IBM Germany within the project "Databases in Server-Workstation Environment".

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

signature file	data base name	delivery	order ono description	order text
s_1 0110101110	o_1 Magnum	Hawaii	105 CRT 107 PC-AT/370 209 Graphic Board	This order is applied within our leasing contract with Hawaii Leasing Corp.
s_2 1001101010	o_2 Kojak	New York	106 PC-Portable 007 5-Finger-Mouse	The FBI is to be charged for this order
s_3 1010010111	o_3 Derrick	Munich	202 64-KB Board 105 CRT	This material shall be bought on credit
s_Q 0010000110	$Q = \{description = 'CRT', order\ text \supseteq \{'leasing'\}\}$			
$R_s = \{o_1, o_3\}$	$R_Q = \{o_1\}$ $_3$ is a false drop			

fig. 1: single-level signature method

To support a faster access, multi-level signature methods are suggested [PFAL80, SACK83]. In the *indexed signature* approach [PFAL80] a signature in the i-th level is created by superimposing (OR-ing) the signatures contained in one page of the (i-1)-th level. On the first level (leaves) we have one signature associated with one object or the object itself. Retrieval is processed by recursively searching the index tree with the query signature as a filter to cut off paths that cannot satisfy the query. In static environments this method provides a good retrieval performance if the file creation is done well [see PFAL80].

However, the office environment is dynamic [CHRI-84]. Therefore we need a method, which provides fast retrieval and which can also be balanced in the presence of insertions, deletions and updates. That is, no periodic reorganisation should be required. In this paper we want to show how dynamic restructuring as in B-trees [BAYE72] can be adopted to the signature approach. Therefore a new appropriate splitting technique is developed.

In the sequel we give a new approach of indexed signatures, called S-tree. The S-tree provides a considerable performance gain and achieves a clustering on a signature basis. In section 2 we show the motivation for the new approach, section 3 gives a definition of the S-tree. Section 4 presents the operations retrieve, insert, delete together with the maintenance procedures. The performance of retrieval is analysed in the following section 5. Section 6 gives some experimental results. In section 7 we outline the clustering feature; section 8 makes some remarks on adaptability. Finally, section 9 summarises the results.

2. Basic Concept of S-Tree

In the dynamic office environment retrieval and insertions are frequent operations. Applying the indexed signature approach here, means to store the signatures in the order the associated objects are inserted. That is, new signatures are entered into the last partially filled node, and ORed to the covering signatures above [cf. PFAL80]. Although [PFAL80] suggests to support the expectation of many insert operations by partially filling the nodes at creation time, performance may decrease because the signature tree is designed as a static structure. Moreover, a frequent processing of a time consuming periodic reorganisation should be avoided in the dynamic office environment. Often new inserted signatures would not be inserted into the appropriate leaf (see fig. 2). Here, "appropriate" means the leaf where similar signatures are stored. "Similar signatures" are signatures with many set *ones* in same positions.

The superimposing of dissimilar signatures causes an increase of set *ones*. For example, consider the four signatures of node B2 in fig. 2, each with three set *ones*. By superimposing these signatures results the first signature in root B1 with seven set *ones*. Consequently, the nodes near the root become less selective and more paths of the tree must be searched for retrieval. Consider the query signature $s(Q) =< 10100010 >$ applied on the tree given in figure 2; all nodes must be accessed.

To avoid such a degeneration we should insert new signatures into leaves where similar signatures were already stored. If the appropriate leaf is already full, we will partition the signatures into two groups according to their similarity and put them into two nodes. This will prevent, that the superimposing of a single node's signatures to obtain the signature of the next level, causes a quick increase in set *ones*. Figure 3 shows such an S-tree, applied on the same object base as fig. 2. Consider the leaf N7; four similar signatures are ORed to form the first signature in node N3, which obtains four *ones*, just a single *one* more. If we apply the same query with $s(Q) =< 10100010 >$ we must search just one short path (N1, N3) to recognise that no match exists in the object base. A successful search is processed faster too, e.g.

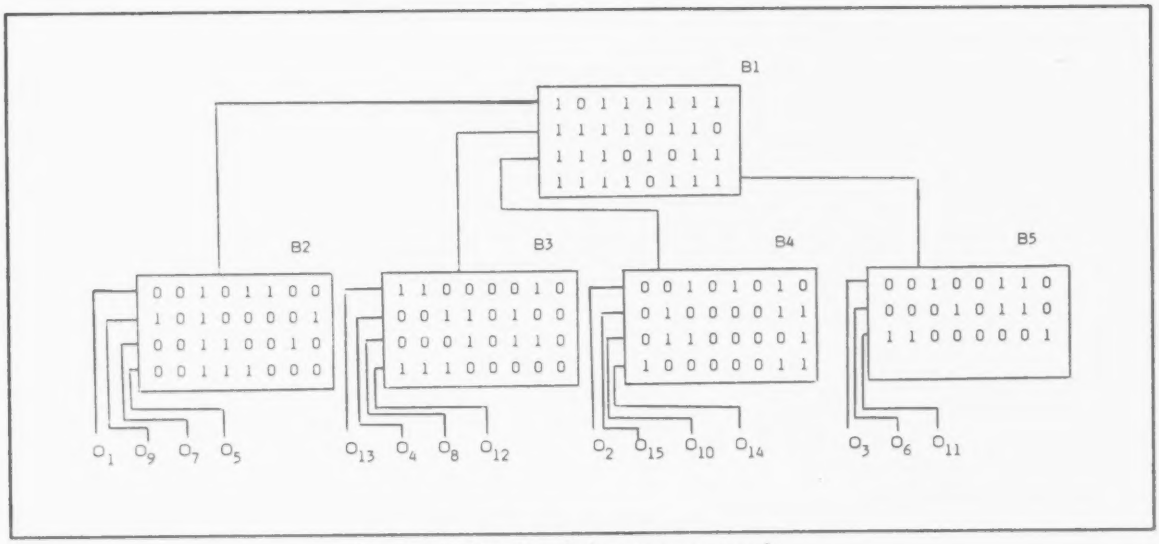


fig. 2: static indexed signature approach

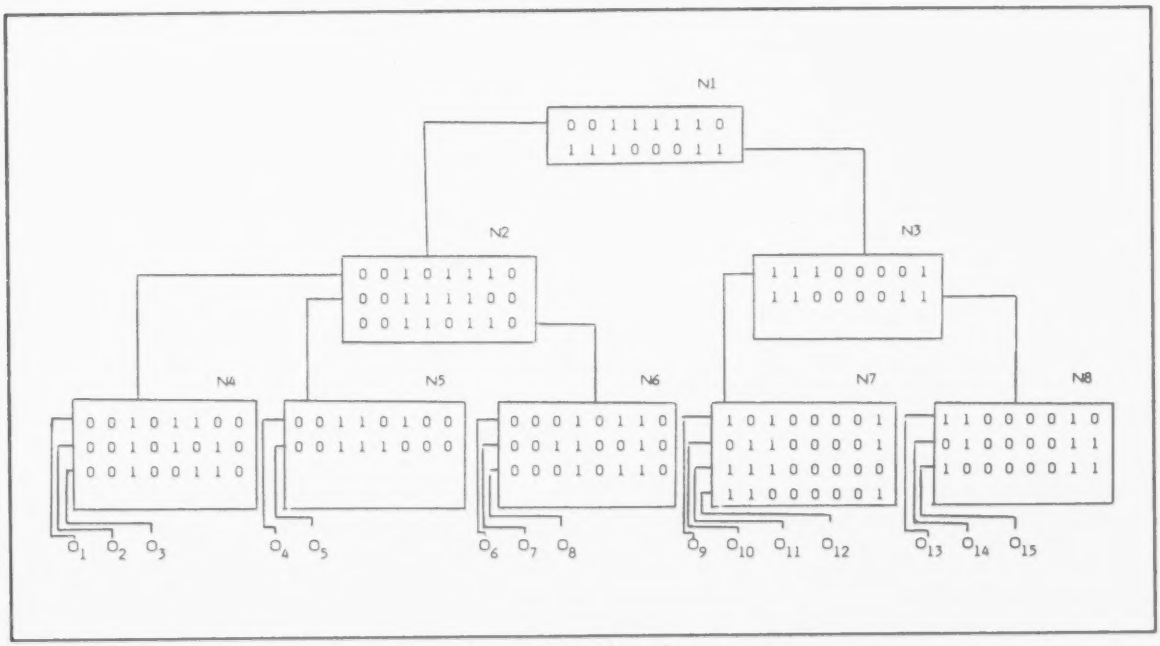


fig. 3: S-tree (4,2,3)

$s(Q) =< 11000010 >$. Moreover, due to the dynamic restructuring, the S-tree needs no periodic reorganisation to keep the performance.

3. Structure of S-Tree

3.1 Definition

Similar to a B^+-tree, an S-tree is a height balanced multiway tree, whose index part is managed like a B-tree [cf. BAYE72]. Each node corresponds to a page. The leaf nodes contain either the objects or object identifiers (*oids*). The former case we call an *immediate S-tree*, the latter a *mediate S-tree*. The leaves of a mediate S-tree contain entries of the form $< s, oid >$ where the object is accessed by the *oid*. The signature s is generated by applying an appropriate hash transformation on the object's attribute values, which maps them into a bit string $s = b_1|b_2|\ldots|b_L$ of fixed length L with $b_i \in \{0,1\}$. A signature in a non-leaf node is defined by superimposing the signatures contained in it's son node (via the signature operator σ). Therefore, entries E in non-leaf nodes have the form $< s, p >$ with the property

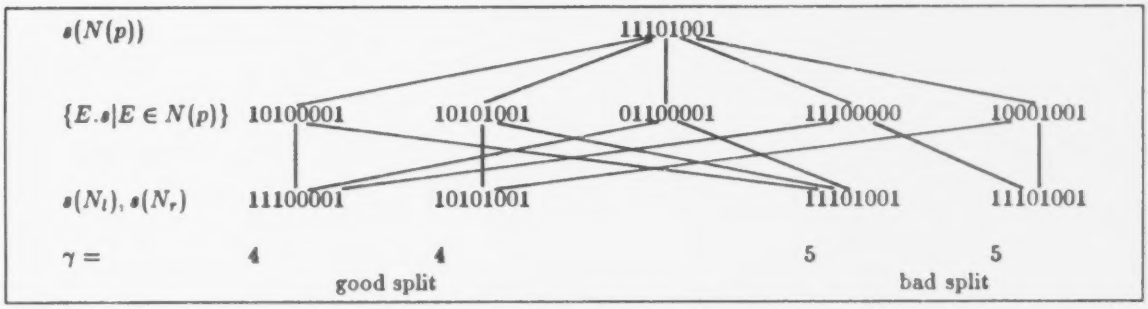


fig. 4: Splitting (K=4,k=2)

[1] $s = s(N(p)) := \sigma(\{E.s|E \in N(p)\}) := \bigvee_{E \in N(p)} E.s$

where $N(p)$ refers to the Node p and $E.s$ denotes the signature component of an entry E. Now we can define a *mediate S-tree* of the type (K, k, h), $K, k, h \in N_0$, with the following properties:

(1) Each path from the root to any leaf has the same length h (height).
(2) The root has at least 2 and at most K sons unless it is a leaf.
(3) Every node except the root has at least k and at most K sons.
(4) The signatures contained in each non-leaf node are minimal w.r.t. [1].

The height h of an S-tree for n objects is at most $\lceil log_k n \rceil - 1$. The minimum number of nodes in an S-tree is $\#_{min}(N) = 1 + 2\sum_{i=0}^{h-2} k^i = 1 + 2\left(\frac{k^{h-1}-1}{k-1}\right)$, the maximum number is $\#_{max}(N) = \sum_{i=0}^{h-1} K^i = \frac{K^h-1}{K-1}$. The minimum space utilisation for all nodes is $\frac{k}{K}$. Contrary to a B-tree, there exists no order of entries within a node. Moreover, due to hash-coding it is possible that the same signature appears multiple times in an S-tree (see fig. 3: the signatures of object O_6 and O_8).

3.2 Problem of Splitting

According to the original B-tree we could choose the minimal branching factor $k = \frac{K}{2}$. However, the entries in the S-tree have not only a guidance function. Since the signatures are abstractions of sets, they contain already some more information about the objects. Due to superimposing, the signatures near the root contain more set *ones* than signatures near the leaves (see fig.3). In the retrieval process, it is not unlikely that nodes with many set *ones* are accessed, without leading to a hit (cf. the first example in section 2). Therefore a more flexible splitting technique is required which allows $1 \leq k \leq \frac{K}{2}$ (cf. analog problem for spatial searching [GUTT84]. To achieve a good split we must pay attention to partition the $K+1$ signatures belonging to one node to split, such that the signature weights

$$\gamma(s) := \sum_{i=1}^{L} b_i$$

of the two new nodes are low. That is, their signatures should have as few set *ones* as possible. Figure 4 illustrates an example for a good and a bad split. Here, the bad split produces not only heavier signatures, but also the same signature pattern. Consequently, retrieval proceeds either on both nodes or on none, i.e. the split does not increase selectivity. Let $s(N_l)$ and $s(N_r)$ be the signature of the left and the right brother node after splitting, we have to solve the following problem:

$$min! \quad \gamma = \gamma(s(N_l)) + \gamma(s(N_r))$$

with the constraint

$$k \leq \#(N_l), \#(N_r) \leq K.$$

However, since up to now no efficient algorithm for that problem is known (to the author), we solve this problem heuristically, and hence possibly suboptimal.

4. Operations of the S-Tree

4.1 Retrieval

Similar to the B-tree the search algorithm descends within the S-tree from the root down to the leaves. However, more than one subtree may be visited recursively [cf. PFAL80]. To demonstrate the retrieval, consider the following query signature $s(Q) =< 00110000 >$ applied on the S-tree of fig. 3. The first signature of the root leads us to his son (N2), because it contains a *one* at the third and the fourth position. Processing proceeds here at the second and third signature. In the leaf N5 we find two match candidates (O_4, O_5), in the leaf N6 we have one (O_7). However, it is possible to gather all object identifiers first, sort them according to page numbers and fetch and check the match candidates in one pass, i.e. without visiting a single page twice.

4.2 Insertion

After inserting a leaf entry in a suitable leaf, the signature of that leaf must be updated. If the leaf signature has changed, this change must be propagated upwards within the S-tree. If the leaf node becomes overfull, it has to be split. Splits propagate upwards too. To find the appropriate leaf node, the CHOOSE function descends in the

```
Initialize NODE := ROOT.NODE; START := ROOT.DEPTH; DEPTH := START; s := s(Q);
Procedure RETRIEVE (START, DEPTH, NODE, s)
Fetch (NODE);
for each entry E in NODE do
    if s ∧ E.s = s
    then if DEPTH < HEIGHT
         then RETRIEVE (START, DEPTH + 1, E.p, s)
         else Check (E.oid)    (* check object and return *)
end;                           (* good drops to response set *)
if PAGE.NEXT(NODE) ≠ nil ∧ DEPTH = START              (* for NEXT option *)
then RETRIEVE (START, DEPTH, PAGE.NEXT(NODE), s); (* see section 8 *)
end; (* end RETRIEVE *)
```

```
Initialize STOP-DEPTH = HEIGHT;
Procedure INSERT (IE, STOP-DEPTH)    (* IE = entry to be inserted *)
DEPTH := ROOT.DEPTH;
if DEPTH = 1 then NODE := CHOOSE (ROOT, DEPTH, STOP-DEPTH, IE.s)
             else NODE := CHOOSE-NEXT (ROOT, DEPTH, STOP-DEPTH, IE.s);
if NODE is full then SPLIT (NODE, DEPTH, IE)
                else begin
                     Add IE to NODE;
                     ADJUST (NODE, DEPTH, IE.s);
                     end;
end; (* end INSERT *)
```

```
Function CHOOSE (NODE, DEPTH, STOP-DEPTH, s) returns NODE varifies DEPTH
Fetch (NODE);
if DEPTH = STOP-DEPTH then CHOOSE := NODE
                      else begin
                           Find entry E with minimal ε(s, E.s);
                           CHOOSE := CHOOSE (E.p, DEPTH + 1, STOP-DEPTH, s);
                           end;
end; (* end CHOOSE *)
```

```
Procedure ADJUST (NODE, DEPTH, s)
Fetch (Parent(NODE));
Find entry E with E.p = NODE;
s := E.s ∨ s;
if E.s ≠ s then begin
                E.s := s;
                if DEPTH > ROOT.DEPTH then ADJUST (Parent(NODE), DEPTH - 1, s);
                end;
end; (* end ADJUST *)
```

S-tree on one single path. This path is selected step by step by applying a similarity measure to the signatures in a node (s) and the signature to be inserted (s'). We may use the *Hamming distance metric*:

$$\delta(s, s') := \gamma(s \vee s') - \gamma(s \wedge s').$$

However, to achieve a high selectivity in each node we need a low weight $\gamma(s(N))$. Therefore it is better to choose that node to proceed which obtains the lowest weight increase. So we define the *weight increase distance* ϵ:

$$\gamma(s \vee s') = \gamma(s) + \epsilon(s, s')$$

$$\epsilon(s, s') := \gamma(s \vee s') - \gamma(s) = \gamma(\neg s \wedge s').$$

```
Procedure SPLIT (NODE, DEPTH, IE)
Add IE to NODE;    (* virtually *)
α := {(Find Entry E with maximal γ(E.s))};          (* find heaviest entry *)
β := {(Find Entry E with maximal ε(s(α), E.s))};    (* find entry with maximal weight increase *)
for each remaining Entry E do
    if #α + #(remaining entries) = k then Assign all remaining entries to α (* stop *);
    if #β + #(remaining entries) = k then Assign all remaining entries to β (* stop *);
    if ε(s(α), E.s) < ε(s(β), E.s) then Assign E to α
                                   else Assign E to β;
end;
Get a fresh page NEWNODE;
Set PAGE.NODE(NEWNODE) to β;
IE :=< ⋁_β E.s, NEWNODE >;
Set PAGE.NODE(NODE) to α;
if NODE ≠ ROOT.NODE then begin
                                   Fetch (Parent(NODE));
                                   Find entry E with E.p = NODE;
                                   E.s := ⋁_α E.s;
                                   NODE := Parent(NODE);
                                   if NODE is full then SPLIT (NODE, DEPTH, IE)
                                                   else begin
                                                        Add IE to NODE;
                                                        ADJUST (NODE, DEPTH, IE.s);
                                                        end;
                                   end;
else if CUT = false then begin
                         Get a fresh page ROOT;
                         Add < ⋁_α E.s, NODE > to ROOT;
                         Add IE to ROOT;
                         HEIGHT := HEIGHT +1
                         end;
                    else begin
                         PAGE.NEXT(NEWNODE) := PAGE.NEXT(NODE);
                         PAGE.NEXT(NODE) := NEWNODE;
                         end;
end; (* end SPLIT *)
```

Notice, the weight increase distance is not commutative and therefore no metric. By applying the weight increase distance, the adjustment propagation is better bound. For refinement we may use the Hamming distance as a secondary strategy.

4.3 Splitting

If a node into which an entry should be inserted is already full, it will be split. That is, the $K+1$ entries should be partitioned into two nodes in such a way that the weight of the two new signatures is low and the Hamming distance between them is high. This reduces the probability that a query requires both nodes to be accessed. Instead of an exhaustive enumeration of all possible partitions, we give a simple heuristic, which needs only linear time.

In the first step the algorithm picks that entry that has the heaviest signature weight. Since this entry would be the heaviest in one of the two partitions too, and since it must be assigned to one of them, we choose this entry as *seed* α. As a second *seed* β we select that entry, which has the maximum number of set *ones* in positions where α has *nulls*, i.e. the entry providing the maximal weight increase to α. This is reasonable since superimposing the signatures to achieve the node's signature never reduces the weight. Thereafter the remaining signatures are assigned to that seed where they achieve the lowest weight increase, in equality to β because of it's lower weight. Further refinements are possible but not very promising because of the minimal node load k.

4.4 Deletion

Together with pure deletions we take updates into consideration that change the objects' signatures. Updates are managed via delete and insert. Deleting entries from an S-tree causes some maintenance to keep the desired tree properties. The original B-tree presents the catena-

```
Procedure DELETE (s, oid)
SEARCHSTOP := false;
SEARCHDEL (ROOT.NODE, ROOT.DEPTH, s, oid, SEARCHSTOP);
for DEPTH = ROOT.DEPTH until HEIGHT do
    if U(DEPTH) ≠ 0 then INSERT (U(DEPTH), DEPTH);
end;
end; (* end DELETE *)
```

```
Procedure SEARCHDEL (NODE, DEPTH, s, oid, SEARCHSTOP)
Fetch (NODE);
repeat for each entry E in NODE
        if s ∧ E.s = s
        then if DEPTH < HEIGHT
             then SEARCHDEL (E.p, DEPTH + 1, s, oid, SEARCHSTOP);
             else if E.oid = oid then begin
                                      Delete entry E;
                                      SEARCHSTOP := true;
                                      end;
until SEARCHSTOP;
if SEARCHSTOP = true and NODE has fewer than k entries and NODE ≠ ROOT
        then begin
             F := Parent(NODE);
             Fetch (F);
             Delete entry E with E.p = NODE in F;
             U(DEPTH) := NODE;
             end;
end; (* end SEARCHDEL *)
```

tion and underflow algorithm for that purpose [BAYE72]. In the approach of [GUTT84] underfull nodes are eliminated. This is propagated upwards, and then the entries with possible subtrees are reinserted at the correct level. Here we adopt the reinsertion strategy because updates and deletions are less frequent than retrieval or insertions, especially in the office environment. The reinsertion incrementally refines the S-tree structure according to adjacency of signatures. Furthermore the efficiency of reinsertion should be comparable the corresponding B-tree algorithms because pages requested are already fetched into primary memory due to the proceeding search [GUTT84].

5. Analysis

To estimate the performance we assume that a partial-match query with t set *ones* should be processed. We give a simple analysis in terms of node fetches (I/O). We make the following assumptions concerning the generation of the signatures:

(a1) each signature has the same signature weight $\gamma < \frac{L}{3}$,

(a2) hash-coding provides an equal distribution of set *ones*.

Notice that assumption (a1) does not fix the coding method. From assumption (a2) follows, that the $\binom{L}{\gamma}$ possible distinct signatures are equally probable. For a signature which is derived by superimposing λ signatures with weight γ, the probability that t prespecified positions contain *ones* is [ROBE79, SACK83]:

$$p(t) = \sum_{j=0}^{t} \frac{(-1)^j \binom{t}{j} \binom{L-j}{\gamma}^{\lambda}}{\binom{L}{\gamma}^{\lambda}}.$$

For the sake of simplicity Roberts derived in [ROBE79] the following approximation:

$$p(t) \approx \left[1 - \left(1 - \frac{\gamma}{L}\right)^{\lambda}\right]^{t}.$$

We can explain the approximation as follows: $\frac{\gamma}{L}$ is the probability that one prespecified bit position contains a *one*. Hence $1 - \frac{\gamma}{L}$ is the probability for a *null*. Raising to power with λ results the probability to keep the *null* by superimposing λ signatures. The correct probability that one prespecified position contains a *one* is 1 minus that term. By raising to power with t the approximation for t prespecified positions is achieved.

In the S-tree one signature in an i-th level node is generated by superimposing the signatures in the corresponding son. Therefore each signature in a node is the result from superimposing a subset of several objects' signatures. At the root this subset has the cardinality

$\lambda(1) = \frac{n}{\beta(1)}$, where n is the number of all objects and $\beta(1)$ describes the expected utilisation of the root. We obtain for depth d

$$\lambda(d) = \frac{n}{\prod_{i=1}^{d} \beta(i)}.$$

We keep the assumption that all L positions are set with equal probability for an estimation purpose, although the combining of similar signatures in deeper levels produces lower weights. For each depth d we obtain the following formula

$$p(t,d) \approx \left[1 - \left(1 - \frac{\gamma}{L}\right)^{\lambda(d)}\right]^{t}.$$

Now the number of matches at depth d is

$$\prod_{i=1}^{d} (\beta(i) \cdot p(t,i)),$$

and each match causes a node access of level $d+1$. Consequently the number of node accesses in the whole S-tree can be estimated by

$$\sum_{d=1}^{h-1} \left(\prod_{i=1}^{d} (\beta(i) \cdot p(t,i)) \right) + 1.$$

6. Experimental Results

We have implemented the S-tree [SEMS86] to compare its performance against the single-level signature method. In this section we present preliminary results of three experiments. Each experiment involves two steps, first building an S-tree of pseudo-random object signatures and second running a set of pseudo-random query signatures against it.

The generation of pseudo-random query signatures is designed independent of the object signatures. Each S-tree is tested with several query signature weights. For each single weight 60 retrieval operations with different signatures are executed and the average number of I/Os is presented. Each set of query signatures contains successful and unsuccessful searches. A special test set containing only successful query signatures requires up to four I/Os more than the presented results for the mixed set.

The following experiment parameters are set. For each experiment the page size is 2 K bytes, the size of a pointer is 4 bytes. Experiment 1 concerns signatures with length 256 bits and weight 40. The minimal node load k is 20, the maximal node load K is 56. Experiment 2 and 3 concern signatures with length 512 and weight 80 resp. 120, $k = 10, K = 30$. The lower part of the presented tables shows the average number of I/Os for the S-tree corresponding to the query weight. For comparison purpose we have computed the number of I/O-operations for scanning the one-level signature file, where the same signatures were stored sequentially and dense, i.e. without gaps. The results here are independent of the weight of the query signature.

In comparison with the single-level signature method the S-tree of experiment 1 achieves a better performance (see table 1). Moreover, performance is improved through the longer signatures. For example, the S-tree based on 256/40 signatures (table 1) needs 152 I/Os for a query with a 10 set *ones* (a quarter weight), while the S-tree with the 512/80 signatures (table 2) requires only 75 I/Os for a corresponding query with 20 set *ones*.

Compared to the single-level method the S-tree with a signature length of 512 bits improves the performance

number of objects	1000	2000	5000	10000
single-level method	18	36	90	179
S-tree				
query weight				
10	18	32	75	152
20	12	19	45	87
30	8	12	28	51
40	6	8	18	32

table 1: I/Os of experiment 1 (256/40,20,56)

number of objects	1000	2000	5000	10000
single-level method	34	67	167	334
S-tree				
query weight				
5	34	65	160	315
10	19	36	90	177
20	9	15	38	75
30	6	10	24	46
40	5	8	19	36
50	4	7	17	32
60	4	7	17	31
70	4	7	16	31
80	4	7	16	30

table 2: I/Os of experiment 2 (512/80,10,30)

number of objects	1000	2000	5000	10000
single-level method	34	67	167	334
S-tree				
query weight				
10	39	74	192	391
20	28	51	130	240
30	20	36	91	172
40	15	26	68	126
50	12	20	52	94
60	10	16	41	74
70	8	13	34	61
80	7	11	28	52
90	6	10	25	47
100	5	9	22	41
110	5	8	21	38
120	5	8	19	36

table 3: I/Os of experiment 3 (512/120,10,30)

even if the signature weight is increased (see table 3). The tables 2 and 3 show that the S-tree performs better for a query weight of 10% of object signature weight or more. The more bits are set to *one* in the query signature the less nodes are accessed. The height of the S-trees in experiment 2 and 3 is three, they increase the storage overhead of single-level signatures by factor 1.4 up to 1.9.

7. Clustering and Refined Retrieval

The immediate S-tree contains the objects in it's leaves. Their signatures may be stored with them, but they can also be computed by demand. To achieve nearly the same branching factor we may define a larger page size for the leaves, because the objects are larger than signatures. This approach groups objects together with similar signatures. Therefore the immediate S-tree realizes a *clustering* according to the hash-coding [cf. SALT78, RIJS79]. Performance is increased because it is probable that more objects of one leaf are match candidates. Due to the split operation, the S-tree keeps the clustering in a highly dynamic environment. There is no extra reorganisation and no extra cluster generation necessary. But the quality of clustering depends on the hash-coding. Here a hash-coding is required which preserves the similarity of the objects.

A second advantage of the S-tree is that it allows a refined retrieval. In the office environment a browsing capability is needed [TSIC85b]. This iterative browsing function can be supported by applying relevance feedback methods [cf. RIJS79, SALT83b] or nearest neighbor search on signature basis. One possibility may be that the user pinpoints those objects in the initial response set which seem to be relevant for him. Thereafter the signatures associated with these selected objects may be ANDed to discover their joint properties. The resulting signature may be applied on the object base for the next search iteration.

Nearest neighbor search on signature basis can be supported by extending the signature test with the Hamming distance or a distance ϱ which allows a certain number of set *one* positions to be *null*. The latter case can be done easily within the S-tree retrieval procedure. We just replace the test condition with $\gamma(s(Q) \wedge E.s) \geq \gamma(s(Q)) - \varrho$. Another refinement is the opportunity to embed a more sophisticated filter [e.g. similar to SALT83a] instead of one single query signature.

8. Adaptability

8.1 Light Queries

A partial-match query is defined as a *light query* if it's signature weight is lower than a given weight g

$$\gamma(s(Q)) < g.$$

In the case of a light query the probability is high, that in a big and high S-tree a great part of the tree has to be searched recursively. The signature of a light query has only a few set *ones* and therefore it is a less selective filter. Hence it is suitable here to search directly the leaves of a mediate S-tree or one level above in the case of the immediate S-tree. To support this, we chain the leaves containing first-level signatures (see fig. 5). This is similar to the one-level signature scan. As anchor we use the new parameter LEAF. To process a light query we call the RETRIEVE procedure with the parameters ($LEAF.DEPTH, LEAF.DEPTH, LEAF.NODE, s(Q)$). That is, the START parameter is equal to the depth of the leaves. Then the second part of the RETRIEVE procedure is processed which searchs sequentially the leaves.

8.2 Cutting the Top of the S-Tree

For very large object bases we cannot exclude that the root and nodes near to it have a heavy signature weight or even $\gamma(s) = L$. In such a case nearly all bits are set to *one* and the selectivity of these nodes is decreased. The reason for this is given in superimposing, heuristical splitting and the minimum node utilisation k. Here it is reasonable to cut off the top of the S-tree. Descending from the root those nodes are discarded, which contain mostly signatures heavier than a given threshold. By chaining the new roots we obtain a forest of S-trees (see fig. 6), which is more suitable for very large data bases. The consequences resulting from that approach are already taken into account in the given algorithms above via the use of the NEXT-pointer. Since we initialise the START parameter with the new depth of the root, the second part of the RETRIEVE procedure provides a search of the forest of S-trees. For insertion, the CHOOSE-NEXT function visits firstly every root and then descends the most suitable S-tree. Finally, in the split process the chaining of roots is controlled by the CUT variable.

9. Summary and Conclusion

Recently, several related approaches have been made. To build a tree of signatures, we could interprete the signatures as binary numbers and apply an ordinary B^+-tree (similar to the zkd-B-tree in [OREN84]). But this would not perfectly utilise the feature that a signature due to hash-coding always describes a set of objects [DADA83]. The approach of [PRAB83] applies a signature for each node of an ordinary B-tree as an additional filter. In contrast to these approaches, we keep signatures as guides on the path and use the descriptive power of signatures to describe sets.

In this paper we have presented a new approach of dynamic indexed signatures for the retrieval of objects in an office information system. Due to heuristical splitting the S-tree is balanced and keeps adjacent signatures together. This increases retrieval performance in most cases. An important and desired feature is that the answer to a more specified query usually is found quicker than the answer to a less specified one. The S-tree im-

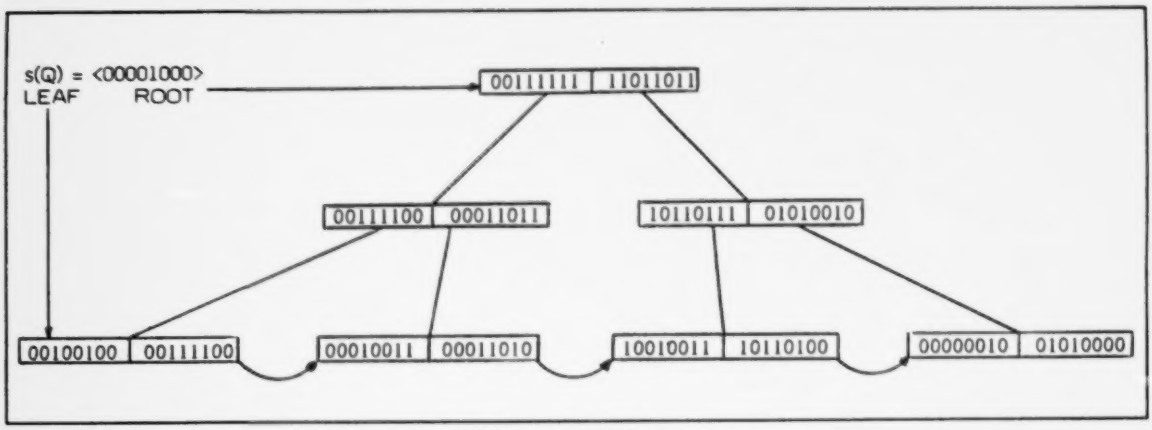


fig. 5: S-tree with light query

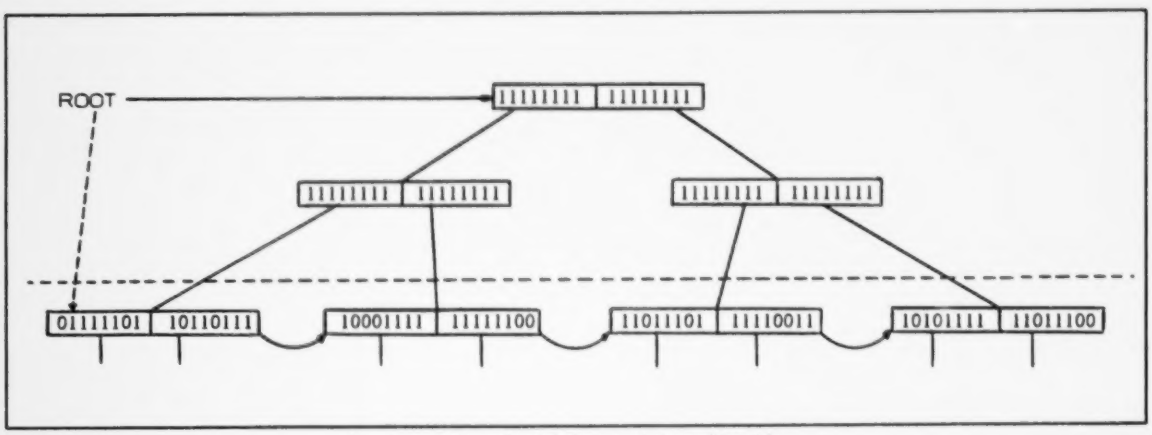


fig. 6: cutting the top of an S-tree (2,1,h)

proves this feature. Moreover, the S-tree supports clustering on a signature basis, since similar signatures are grouped together. This could be the basic idea for applying more refined retrieval methods. Because of the S-tree's adaptability it seems usable in many office environments. Therefore, we have started an implementation to prove it's quality in some experiments [SEMS86], and plan an integration into a data base system for advanced applications (DASDBS [DEPP85] or AIM [DADA86]).

Acknowledgements

Prof. H.-J. Schek provided many interesting suggestions and thoughtful comments. His help is gratefully acknowledged. Moreover the author wishes to thank F. Gebhard (GMD), P. Pistor (IBM HDSC), V. Obermeit, H.-B. Paul, M.H. Scholl, W. Waterfeld, G. Weikum, T. Semsroth and A. Wolf, who have contributed to clarifying discussions of the material. Thanks also to Lis Klinger who helped to preparate the figures.

References

BAYE72 Bayer, R., McCreight, E.M.: Organisation and Maintenance of Large Ordered Indexes, Acta Informatica 1972, pp. 173 - 189.

CHRI84 Christodoulakis, S. et al.: Development of a Multimedia Information System for an Office Environment, Proc. VLDB 1984, pp. 261 - 271.

DADA83 Dadam, P., Pistor, P., Schek, H.-J.: A Predicate oriented Locking Approach for Integrated Information Systems, in: Mason, R.E.A. (ed.): Information Processing 83, Proc. IFIP 1983, pp. 763 - 768.

DADA86 P. Dadam, K. Kuespert, F. Andersen, H. Blanken, R. Erbe, J. Guenauer, V. Lum, P. Pistor, G. Walch: A DBMS Prototype to Support Extended NF^2 Relations: An Integrated View on Flat Tables and Hierarchies, accepted for SIGMOD 1986.

DEPP85 Deppisch, U., Obermeit, V., Paul, H.-B., Schek, H.-J., Scholl, M.H., Weikum, G.: The Storage Subsystem of a Data Base Kernel System (in German), Proc. GI Conf. on Database Systems for Office Automation, Engineering and Scientific Applications 1985, pp. 421 - 440; English version available as Technical Report DVSI-1985-T1, TU Darmstadt, 1985.

FALO84 Faloutsos, C., Christodoulakis, S.: Signature Files: An Access Method for Documents and its Analytical Performance Evaluation, TOOIS 1984, pp. 267 - 288.

FALO85a Faloutsos, C.: Access Methods for Text, Computing Surveys 1985, pp. 49 - 74.

FALO85b Faloutsos, C.: Signature files: Design and performance comparison of some signature extraction methods, Proc. SIGMOD 1985, pp. 63 - 82.

FALO85c Faloutsos, C., Christodoulakis, S.: Design of a Signature File Method that Accounts for Non-Uniform Occurence and Query Frequencies, Proc. VLDB 1985, pp. 165 - 170.

GIBB83 Gibbs, S., Tsichritzis, D.: A Data Modelling Approach for Office Information Systems, TOOIS 1983, pp. 299 - 319.

GUTT84 Guttman, A.: R-Trees: A dynamic index structure for spatial searching, Proc. SIGMOD 1984, pp. 47 - 57.

HARR71 Harrison, M.C.: Implementation of the Substring Test by Hashing, CACM 1971, pp. 777 - 779.

OREN84 Orenstein, J.A., Merrett, T.H.: A Class of Data Structures for Associative Searching, Proc. PODS 1984, pp. 181 - 190.

PFAL80 Pfaltz, J.L., Berman, W.J., Cagley, E.M.: Partial-Match Retrieval using Indexed Descriptor Files, CACM 1980, pp. 522 - 528.

PRAB83 Prabhakar, T.V., Sahasrabuddhe, H.V.: Signature Trees - A Data Structure for Index Organisation, Proc. IEEE Conf. on Systems, Man and Cybernetics 1983, pp. 1145 - 1147.

RABI84 Rabitti, F., Ziska, J.: Evaluation of Acces Methods to Text Documents in Office Systems, in: Rijsbergen, van, C.J. (ed.): Proc. Research and Development in Information Retrieval 1984, pp. 21 - 40.

RABI85 Rabitti, F.: A Model for Multimedia Documents, in: TSIC85a, pp. 227 - 250.

RIVE76 Rivest, R.L.: Partial-Match Retrieval Algorithms, SIAM Journal of Computing 1976, pp. 19 - 50.

RIJS79 Rijsbergen, van, C.J.: Information Retrieval, 2nd ed., London: Butherworth 1979.

ROBE79 Roberts, C.S.: Partial-Match Retrieval via the Method of Superimposed Codes, Proc. IEEE 1979, pp. 1624 - 1642.

SACK83 Sacks-Davis, R., Ramamohanarao, K.: A two-level superimposed coding scheme for Partial Match Retrieval, Information Systems 1983, pp. 273 - 280.

SALT78 Salton, G., Wong, A.: Generation and Search of Clustered Files, TODS 1978, pp. 321 - 346.

SALT83a Salton, G. Fox, E.A., Wu, H.: Extended Boolean Information Retrieval, CACM 1983, pp. 1022 - 1036.

SALT83b Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval, New York: McGraw-Hill 1983.

SCHE77 Schek, H.-J.: Tolerating Fussiness in Keywords by Similarity Searches, Kybernetes 1977, pp. 175 - 184.

SCHE78 Schek, H.-J.: The Reference String Access Method and Partial Match Retrieval, IBM Scientific Center Report TR 77.12.008; partly contained in: The Reference String Indexing Method, in: Bracchi, G., Lockemann, P.C. (eds.): Proc. Information System Methodology 1978, Springer LNCS 65, pp. 432 - 459.

SCHE81 Schek, H.-J.: Methods for the Administration of Textual Data in Data Base Systems, in: Oddy, R.N. et al. (eds.): Information Retrieval Research, London: Butherworth 1981, pp. 218 - 235.

SCHE82 Schek, H.-J., Pistor, P.: Data Structures for an Integrated Data Base Management and Information Retrieval System, Proc. VLDB 1982, pp. 197 - 207.

SENS86 Semsroth, T.: Experimental Studies on S-Trees, Master Thesis, TU Darmstadt, 1986, forthcoming.

TSIC85a Tsichritzis, D. (ed.): Office Automation, Berlin, Heidelberg: Springer 1985.

TSIC85b Tsichritzis, D., Christodoulakis, S., Lee, A., Vandenbroek, J.: A Multimedia Filing System, in: TSIC85a, pp. 43 - 65.

Improved Hierarchical Bit-Vector Compression in Document Retrieval Systems

Y. Choueka[1,2], A.S. Fraenkel[3], S.T. Klein[3], E. Segal[1]

[1] Inst. for Information Retrieval and Computational Linguistics (IRCOL) — The Responsa Project
[2] Department of Mathematics and Computer Science, Bar-Ilan University, Ramat Gan, Israel
[3] Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot 76100, Israel

Part of this work was done while the second and third authors were partially affiliated with IRCOL

ABSTRACT

The "concordance" of an information retrieval system can often be stored in form of bit-maps, which are usually very sparse and should be compressed. Hierarchical bit-vector compression consists of partitioning a vector v_i into equi-sized blocks, constructing a new bit-vector v_{i+1} which points to the non-zero blocks in v_i, dropping the zero-blocks of v_i, and repeating the process for v_{i+1}. We refine the method by pruning some of the tree branches if they ultimately point to very few documents; these document numbers are then added to an appended list which is compressed by the prefix-omission technique. The new method was thoroughly tested on the bit-maps of the Responsa Retrieval Project, and gave a relative improvement of about 40% over the conventional hierarchical compression method.

1. Motivation and Introduction

In some full-text retrieval systems, an inverted file, also called "concordance", is constructed that contains, for every different word W in the data base, a list $L(W)$ of all the documents in which W occurs. In order to find all the documents that contain the words A and B, one has to intersect $L(A)$ with $L(B)$. Usually, A and B stand for families of words A_i and B_j, each family consisting of terms which are considered synonymous for the given query. In this case one has to perform

$$\left(\bigcup_i L(A_i)\right) \bigcap \left(\bigcup_j L(B_j)\right)$$

which can seriously affect the response time in an online retrieval system when the number of involved sets and their sizes are large.

A different approach might be to replace the concordance of a system with l documents by a set of *bit-maps* of equal length l. For every different word W in the data base a bit-map $B(W)$ is constructed, the i-th bit of which is 1 if and only if W occurs in the i-th document, in some fixed ordering of the documents. Processing queries reduces here to performing logical operations with bit-strings, which is easily done on most machines. Davis & Lin [2] were apparently the first to propose the use of bit-maps for secondary key retrieval.

It would be wasteful to store the bit-maps in their original form, since they are usually very sparse (the great majority of the words occur in very few documents). In addition to the savings in secondary storage space, the processing time can be improved by compressing the maps, as more information can be read in a single input operation, thus reducing the total number of I/O accesses. We are interested in a coding pro-

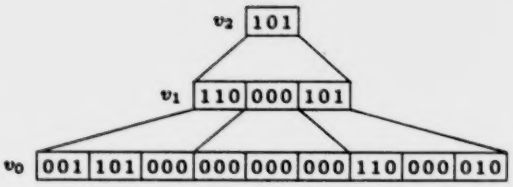
Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

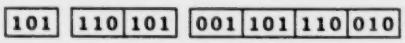
cedure which reduces the space needed to store long sparse binary strings without losing any information; in fact, there should be a simple algorithm which, given the compressed string, can reconstruct the original one. Schuegraf [9] proposes to use *run-length coding* for the compression of sparse bit-vectors, in which a string of consecutive zeros terminated by a one (called a *run*) is replaced by the length of the run. A sophisticated run-length coding technique can be found in Teuhola [10], and various other variants are discussed in Nevalainen, Jakobsson & Berg [8]. Jakobsson [5] suggests to partition each vector into k-bit blocks, and to apply Huffman coding on the 2^k possible bit-patterns. In Fraenkel & Klein [4], the latter method is extended, incorporating run-length coding of blocks consisting only of zeros.

In this paper we concentrate on *hierarchical bit-vector compression*: let us partition the original bit-vector v_0 of length l_0 bits into k_0 equal blocks of r_0 bits, $r_0 \cdot k_0 = l_0$, and drop the blocks consisting only of zeros. The resulting sequence of non-zero blocks does not allow the reconstruction of v_0, unless we add a list of the indices of these blocks in the original vector. This list of up to k_0 indices is kept as a binary vector v_1 of $l_1 = k_0$ bits, where there is a 1 in position i if and only if the i-th block of v_0 is not all zero. Now v_1 can further be compressed by the same method.

In other words, a sequence of bit-vectors v_j is constructed, each bit in v_j being the result of ORing the bits in the corresponding block in v_{j-1}. The procedure is repeated recursively until a level t is reached where the vector length reduces to a few bytes, which will form a single block. The compressed form of v_0 is then obtained by concatenating all the nonzero blocks of the various v_i, while retaining the block-level information. Decompression is obtained simply by reversing these operations and their order. We start at level t, and pass from one level to the next by inserting blocks of zeros into level $j-1$ for every 0-bit in level j.



(a) Original vector and two derived levels



(b) Compressed vector

Figure 1: *Hierarchical Bit-vector Compression*

Figure 1 depicts an example of a small vector v_0 of 27 bits and its derived levels v_1 and v_2, with $r_i = 3$ for $i = 0, 1, 2$ and $t = 2$. The sizes r_j of the blocks are parameters and can change from level to level for a given vector, and even from one word of the database to another, although the latter is not practical for our applications. Because of the structure of the compressed vector, we call this the TREE-method, and shall use in our discussion the usual tree-vocabulary: the *root* of the tree is the single block on the top level, and for a block x in v_{j+1} which is obtained by ORing the blocks $y_1, \ldots, y_{r_j}$ of v_j, we say that x is the *father* of the non-zero blocks among the y_i.

The TREE-method was proposed by Wedekind & Härder [12]. It appears also in Vallarino [11], who used it for two-dimensional bit-maps, but only with one level of compression. In Jakobsson [6], the parameters (block size and height of the tree) are chosen assuming that the bit-vectors are generated by a memoryless information source, i.e., each bit in v_0 has a constant probability p_0 for being 1, independently from each other. However, for bit-maps in information retrieval systems, this assumption is not very realistic a priori, as adjacent bits often represent documents written by the same author; there is a positive correlation for a word to appear in consecutive documents, because of the specific style of the author or simply because such documents often treat the same or related subjects. In our approach, the parameters are first restricted so as to simplify the computer

processing of r_j-bit blocks. From among this restricted set P of parameters, we select those which yield maximal compression.

The experiments were run on the bit-maps which were constructed at the Responsa Retrieval Project (see for example Choueka [1] or Fraenkel [3]) of about 40 million Hebrew and Aramaic words. They showed, a posteriori, that the compression does not vary much with the different elements of P. Thus any choice of parameters from P will do, so the method is efficient also for dynamically changing data bases.

In the next section we suggest some improvements to the TREE-method. We then report in Section 3 on the experiments which led to the parameter setting. Comparison of the results with those obtained on randomly generated bit-vectors shows that the technique is specially well adapted to bit-maps of document retrieval systems. The new algorithm is compared with other methods in the final Section 4.

2. Improvements to method TREE

We first remark that the hierarchical method does not always yield real compression. Consider for example a vector v_0 for which the indices of the 1-bits are of the form ir_0 for $i \leq l_0/r_0$. Then there are no zero-blocks (of size r_0) in v_0, moreover all the bits of v_i for $i > 0$ will be 1, so that the whole tree must be kept. Therefore the method should be used only for sparse vectors.

In the other extreme case, when v_0 is very sparse, the TREE-method may again be wasteful: let $d = \lceil \log_2 l_0 \rceil$, so that a d-bit number suffices to identify any bit-position in v_0. If the vector is extremely sparse, we could simply list the positions of all the 1-bits, using d bits for each. This is in fact the inverse of the transformation performed by the bit-vectors: basically, for every different word W of the database, there is one entry in the inverted file containing the list of references of W, and this list is transformed into a bit-map; here we change the bit-map back into its original form of a list.

A small example will illustrate how the bijection of the previous paragraph between lists and bit-maps can be used to improve method TREE. Suppose that among the $r_0 \cdot r_1 \cdot r_2$ first bits of v_0 only position j contains a one. The first bit in level 3, which corresponds to the OR-ing of these bits, will thus be set to 1 and will point to a sub-tree consisting of three blocks, one on each of the lower levels. Hence in this case a single 1-bit caused the addition of at least $r_0 + r_1 + r_2$ bits to the compressed map, since if it were zero, the whole sub-tree would have been omitted. We conclude that if $r_0 + r_1 + r_2 \geq d$, it is preferable to consider position j as containing zero, thus omitting the bits of the sub-tree, and to add the number j to an appended list L, using only d bits. This example is readily generalized so as to obtain an optimal partition between tree and list for every given vector, as will now be shown.

We define l_j and k_j respectively as the number of bits and the number of blocks in v_j, for $0 \leq j \leq t$. Note that $r_j \cdot k_j = l_j$. Denote by $T(i,j)$ the sub-tree rooted at the i-th block of v_j, with $0 \leq j \leq t$ and $1 \leq i \leq k_j$. Let $S(i,j)$ be the size in bits of the compressed form of the sub-tree $T(i,j)$, i.e., the total number of bits in all the non-zero blocks in $T(i,j)$, and let $N(i,j)$ be the number of 1-bits in the part of the original vector v_0 which belongs to $T(i,j)$.

During the bottom-up construction of the tree these quantities are recursively evaluated for $0 \leq j \leq t$ and $1 \leq i \leq k_j$ by:

$$N(i,j) = \begin{cases} \text{number of 1-bits in block } i \text{ of } v_0 \\ \qquad \text{if } j = 0, \\ \sum_{h=1}^{r_j} N\big((i-1)r_j + h, j-1\big) \\ \qquad \text{if } j > 0; \end{cases}$$

$$S(i,j) = \begin{cases} 0 \quad \text{if } j = 0 \text{ and} \\ \qquad T(i,0) \text{ contains only 0's}, \\ r_0 \quad \text{if } j = 0 \text{ and} \\ \qquad T(i,0) \text{ contains a 1-bit}, \\ \sum_{h=1}^{r_j} S\big((i-1)r_j + h, j-1\big) \\ \qquad \text{if } j > 0. \end{cases}$$

At each step, we check the condition

$$d \cdot N(i,j) \leq S(i,j). \qquad (1)$$

If it holds, we prune the tree at the root of $T(i,j)$, adding the indices of the $N(i,j)$ 1-bits to the list L, and setting then $N(i,j)$ and $S(i,j)$ to zero. Hence the algorithm partitions the set of 1-bits into two disjoint subsets: those which are compressed by the TREE-method and those kept as a list. In particular, if the pruning action takes place at the only block of the top level, there will be no tree at all.

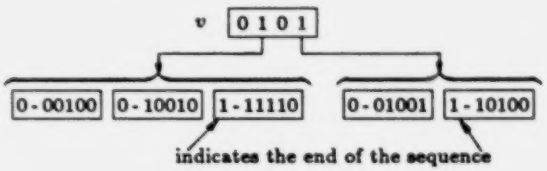
Note that in case of equality in (1), we execute a pruning action although a priori there is no gain. However, since the number of 1-bits in v_j is thereby reduced, this may enable further prunings in higher levels, which otherwise might not have been done.

We now further compress the list L (of indices of 1-bits which were "pruned" from the tree) using the *prefix-omission* technique. It consists of storing a common prefix of several consecutive words only once, and is usually applied to the compression of dictionaries. This can be adapted to the compression of a list of d-bit numbers: we choose an integer $c < d-1$ as parameter, and form a bit-map v of $k = \lceil l_0/2^c \rceil$ bits, where bit i, for $0 \leq i < k$, is set to 1 if and only if the integer i occurs in the $d-c$ leftmost bits of at least one number in L. Thus a 1-bit in position i of v indicates that there are one or more numbers in L in the range $[i2^c, (i+1)2^c - 1]$. For each 1-bit in v, the numbers of the corresponding range can now be stored as relative indices in that range, using only c bits for each, and an additional bit per index serving as flag, which identifies the last index of each range. Further compression of the list L is thus worthwhile only if

$$d \cdot |L| \; > \; k + (c+1)|L|. \qquad (2)$$

The left hand side of (2) corresponds to the number of bits needed to keep the list L uncompressed. Therefore this secondary compression is justified only when the number of elements in L exceeds $k/(d-c-1)$.

For example, for $l_0 = 128$ and $c = 5$, there are 4 blocks of 2^5 bits each; suppose the numbers in L are 36, 50, 62, 105 and 116 (at least five elements are necessary to justify further compression). Then there are three elements in the second block, with relative indices 4, 18 and 30, and there are two elements in the fourth block, with relative indices 9 and 20, the two other blocks being empty. Thus the following information would be kept:



Finally we get even better compression by adapting the cut-off condition (1) dynamically to the number of elements in L. During the construction of the tree, we keep track of this number and as soon as it exceeds $k/(d-c-1)$, i.e., it is worthwhile to further compress the list, we can relax the condition in (1) to

$$(c+1) \cdot N(i,j) \leq S(i,j), \qquad (3)$$

since any index which will be added to L, will use only $c+1$ bits for its encoding.

In fact, after recognizing that L will be compressed, we should check again the blocks already handled, since a sub-tree $T(i,j)$ may satisfy (3) without satisfying (1). Nevertheless, we have preferred to keep the simplicity of the algorithm and not to check again previously handled blocks, even at the price of losing some of the compression efficiency. Often, there will be no such loss, since if we are at the top level when $|L|$ becomes large enough to satisfy (2), this means that the vector v_0 will be kept in its entirety as a list. If we are not at the top level, say at the root of $T(i,j)$ for $j < t$, then all the previously handled trees will be reconsidered as part of larger trees, which are rooted on the next higher level. Hence it is possible that the sub-tree $T(i,j)$, which satisfies (3) but not (1) (and thus was not pruned at level j), will be removed as part of a larger sub-tree rooted at level $j+1$.

The method of dynamically pruning the tree, forming a list L and compressing the latter using prefix omission, will henceforth be called the PRUNE-method.

3. Experimental Results

3.1 Setting the parameters

The above method was tested on the bit-maps of the Responsa data base of $l_0 = 42272$ documents; accordingly, $d = 16$. To allow for easy computer manipulation, we decided to use only blocks of one, two or four bytes, so that the possible variants had to verify

$$\log_2 r_i \in \{3,4,5\} \quad \text{and} \quad \sum_{i=0}^{t} \log_2 r_i = 16.$$

There are 24 such variants, which are listed below in Table 1. Each variant is characterized by a $(t+1)$-tuple $(a_0,\ldots,a_t)$, with $t = 3$ or $t = 4$, such that $a_i = r_i/8$ is the number of bytes in the blocks on level i, for $0 \le i \le t$.

Table 1: *Block sizes corresponding to various possibilities of hierarchical compression*

11112	1144	1441	2222	4141
11121	1224	2124	2421	4212
11211	1242	2142	2421	4221
12111	1414	2214	4114	4411
21111	1422	2241	4122	

For secondary compression of the list, the parameter c was set to 7, so that one byte was used for any number added into list L, instead of two bytes before L had been compressed. The length k of the bit-vector v used to compress L was therefore $\lceil 42272/128 \rceil = 331$, but we chose a 50-byte block ($k = 400$) as was done at the Responsa Project to facilitate updating in case of growth of the data base in the future. Therefore the condition for further compressing the list L was $|L| > 50$, and the number of bytes needed to store the list L was $\min(2|L|, 50 + |L|)$.

In order to compare the 24 block-size patterns, a test was run on a sample of 1040 bit-maps with different frequencies of 1-bits. The following statistics were collected for each map:

1. the *compression factor* (CF) for each method, which is defined as the ratio of the size of the original map to the size of the compressed map;
2. the block-size pattern giving maximal compression;
3. the difference in compression efficiency between the best and the worst pattern.

The test showed that:

(a) For most vectors with up to 300 1-bits there was no tree, only a list of indices.

(b) The best parameter pattern changed very slightly with the different maps and was almost constant for maps with the same 1-bit frequency. In 91% of the sample the optimum was achieved with 5 levels ($t = 4$), and among the optimal 5-level trees, 54% used the pattern 11112.

(c) The compression factor was a non-increasing function of the number of 1-bits in the bit-map. For 18000 1-bits or more there was practically no compression, on the contrary, there were a few examples for which the optimal tree used more storage than the original bit-string.

(d) Using the best pattern for each of the 1040 maps of the sample, the average CF was about 16.7. This is an estimation based on the sample of 1040 bit-maps and taking into account the appropriate weight for each range of frequencies, e.g., the maps with up to 300 1-bits form about 80% of the file to be compressed.

(e) The total difference in compression between choosing the best or worst block-size patterns was very small, about 2.6% on the average.

As a consequence of (a), we decided to restrict ourselves to the 56588 different words which appear more than 70 times in the current Responsa data base. From (e) we concluded that it is not worthwhile to search for the optimal method, and that we could choose a constant pattern for all the maps, which sim-

plifies both the compression and decompression process. The best variant was that corresponding to the pattern 11112, but we preferred the variant with pattern 2222, since the additional amount of storage needed was only about half a percent, on the other side the uniformity of the latter variant and the fact that it used only three levels of compression permitted savings in the time complexity of the decompression routine.

3.2 Comparison with randomly generated bit-vectors

If we assume that each bit in v_0 has a constant probability for being 1, independently from each other, the expected size of the compressed vector obtained by the TREE-method can be evaluated as follows. Let p_i be the probability of a bit in level i to be 1, then for $0 \leq i \leq t$

$$\begin{aligned} p_i &= 1 - (1 - p_{i-1})^{r_{i-1}} \\ &= 1 - (1 - p_0)^{\prod_{j=0}^{i-1} r_j}, \end{aligned}$$

where we define $\prod_{j=0}^{-1} r_j = 1$ for convenience. Since for each block which will be kept at level i, there is a 1-bit in level $i+1$, the expected number of (non-zero) blocks at level i is $k_i p_{i+1}$ for $i < t$. Thus the expected number of bits we keep in level i is $k_i p_{i+1} r_i = l_i p_{i+1}$, which holds also for $i = t$ if we define $p_{t+1} = 1$. Since

$$l_i = k_{i-1} = \frac{l_{i-1}}{r_{i-1}} = \frac{l_0}{\prod_{j=0}^{i-1} r_j} \qquad \text{for} \quad 0 \leq i \leq t,$$

we can express the expected size of the compressed tree (in bits) as:

$$\sum_{i=0}^{t} l_i p_{i+1} = l_t + l_0 \cdot \sum_{i=0}^{t-1} \frac{1 - (1 - p_0)^{\prod_{j=0}^{i} r_j}}{\prod_{j=0}^{i-1} r_j}. \quad (4)$$

Now the parameters in (4) are set as $l_0 = 42272$, $t = 3$, $r_j = 16$ for $0 \leq j \leq 3$, and p_0, the average frequency of the 1-bits in our sample of Responsa maps, is 0.00817. This yields a theoretical CF of 5.5 for the independent model, whereas the test on the Responsa maps showed a CF of 10.5. The difference should be credited to the inter-dependence of adjacent bits, which tends to produce more "compact" initial vectors, and thereby enhance compressibility.

In order to compare the random model with the Responsa maps also for the PRUNE-method, we decided to use computer simulation of random vectors. To permit fair comparison, the random maps were constructed with the same length (5284 bytes) and with a density-distribution similar to that of the Responsa maps. Toward this end the 56588 Responsa maps were partitioned into 101 classes corresponding to different ranges of the number of 1-bits in the maps. The number of generated maps was taken to be 10% of the number of the Responsa maps for each class. For each map a number K was chosen uniformly from the range, then K "random" bits were set to 1. In fact, the correct method of constructing the maps consists of deciding for every bit with probability p_0 if it should be set to 1, yielding an expected number of 1-bits equal to $p_0 l_0$. We preferred the first mentioned method which is more economical to implement and introduced only a slight error, the probability of a given bit to be set to 1 being $1 - (1 - 1/l_0)^K \simeq K/l_0 = p_0$, since l_0 is large and K is relatively small (there are very few maps with large K). The random numbers were generated following Knuth [7].

The algorithm was applied to the complete file of 56588 Responsa maps, yielding an average CF of 17.5, which is better than expected from the test on the small sample. For the file of 5664 randomly generated maps, the CF was only 15.9. Thus, using the PRUNE-method, compression is still better for the Responsa maps than for the random maps, but the difference in compression is not as striking as for method TREE. As was pointed out earlier, the better results on information retrieval maps are due to the fact that the 1-bits appear in a more clustered form than expected by a model of independent bit-positions; hence for the randomly generated maps, the number of blocks at the lowest level which contain at least one 1-bit, will be larger. However, precisely these isolated bits are the main target of the pruning action. Therefore the improvement should be larger for random than for clustered maps, when both have the same 1-bit density.

3.3 Other Variants

The first variant we have experimented with, was to try to complement the bit-vector before its compression for bit-vectors with more than $l_0/2$ 1-bits. However the test showed that this was almost never worthwhile, even not for the most frequent words. An explanation of this fact could be that although the occurrence of a word W in a document may be strongly related to its occurrence in adjacent documents, this seems not always to be true as to the *absence* of W from consecutive documents. Thus complementing the vectors probably destroyed their clustered appearance.

Another variant was to try to reorganize the vectors into an even more clustered form. This obviously cannot be done simultaneously for all the vectors in an optimal, yet still efficient manner. Therefore the documents were simply arranged by decreasing size, so that reorganizing was in fact applying a fixed permutation on the original bit-vectors. The idea was that the probability of a word to appear is greater for a large document than for a small one. By concentrating the bit-positions corresponding to large documents in the same area of the vector, we expected a migration of the 1-bits towards this area. An indication that such a migration indeed happened was obtained by inspecting r_0, the block-size on the lowest level of the tree, which was constructed according to the optimal pattern among those of Table 1, for each map of the sample: the larger the clusters of 1-bits in the vector, the larger r_0 will be in the optimal pattern. Only for 12.7% of the sample of original maps, r_0 for the optimal pattern was larger than one byte, whereas for the permuted maps, this number increased to 57.4%.

The compression results however were rather disappointing: not a single example in our sample gave an improvement after the rearrangement; on the average, the CF decreased by 10.4%. We conclude that compression is enhanced not as much by the existence of a few large clusters, but more by the existence of many small ones. For the original vector, the documents are grouped by author, and for each author, ordered by topic, which yields many small clusters. This order is completely destroyed by the permutation, as there is almost no correlation between the size of a document and its content. The reorganization splits the clusters into single bits which are scattered throughout the vector more or less randomly. Although clusters will eventually form (in particular at the beginning of the vector, which corresponds to the larger documents), there will be also a large number of single bits and the overall compression efficiency decreases.

4. Comparison with other methods

One of the best known methods for bit-map compression is the run-length coding technique, which is borrowed from image compression. One often uses a fixed number b of bits to encode the length of a run. This is practically equivalent to the method which keeps a bit-map in form of a list of the indices of its 1-bits; instead of these indices, we store the *increments* between them. In our experiments on the Responsa maps, the average distance between 1-bits (which is the average length of runs of zeros) was 287.2 bits, but there were also very long runs, so we chose $b = 16$ to encode the run-lengths.

As to Jakobsson's [5] method of using Huffman codes for the 2^k possible k-bit blocks, any codeword has at least one bit, so we can never expect a better CF than k. On the other hand, k cannot be chosen too large since the number of different codewords to be generated, and the size of the encode and decode tables grow exponentially with k. We chose $k = 8$ to facilitate computer manipulation.

The best method in [4] combines Huffman coding and run-length coding. As in [5], the vector is partitioned into k-bit blocks, then the possible lengths of runs of 0-blocks are partitioned into classes C_i, containing run-lengths h which satisfy $2^{i-1} \le h < 2^i$, for $1 \le i \le \lfloor \log_2(l_0/k) \rfloor$. The $2^k - 1$ non-zero block-patterns together with the classes C_i are then assigned Huffman codes; a run of length h belonging to class C_i is encoded by the codeword for C_i, followed by the $i-1$ rightmost bits of the binary representation of $h - 2^{i-1}$. Again, $k = 8$ was chosen and we had classes C_1 to C_{13}.

The compression results are summarized in Table 2. There is one line for each technique; the methods from [5] and [4] are respectively entitled "Huffman coding" and "Huffman+Run-length". The middle column gives the compression factor as computed on the file of Responsa maps, the right hand column shows for each method how much relative additional savings (in percentage) can be obtained when it is replaced by the PRUNE-method. Note that this can be negative, since the method in [4] yields a higher CF. But the difference is small, and on the other hand, the latter method is more complicated; therefore decoding will be time-consuming, so that the new algorithm may be preferable in some applications even to the methods of [4].

Table 2: *Comparison of various methods of bit-vector compression*

	CF	Relative Improvement of method PRUNE
TREE	10.53	39.7%
Run-length	7.65	56.2%
Huffman coding	6.57	62.3%
Huffman+Run-length	18.77	-7.6%
PRUNE	17.45	–

Another criterion for the comparison of the compression methods is their sensitivity to errors. For run-length coding and Huffman coding, a single incorrectly transmitted bit usually renders the tail of the encoded string following the error useless. The combined Huffman+Run-length method is slightly more robust than either of its two components. For the hierarchical methods, if a single parity change error occurs on the lowest level of the tree (and this is usually the major part of the compressed map), there is no damage other than the affected bit itself. If the error occurs in v_1 (the first level of compression), its effect will be like for Huffman or run-length codes: a suffix of the decompressed map will be garbled. Only errors in higher levels may produce completely false vectors, but these levels form only a small part of the compressed file. Using a few additional bytes per map, we can introduce partial error-detection: all we have to add are the lengths of each level in the compressed map. Since the length of level i is determined by the number of 1-bits in level $i+1$, a single error in the higher levels can be detected.

REFERENCES

[1] **Choueka Y.**, Full text systems and Research in the Humanities, *Computers and the Humanities* **XIV** (1980) 153–169.

[2] **Davis D.R., Lin A.D.**, Secondary key retrieval using an IBM 7090–1301 system, *Comm. of the ACM* **8** (1965) 243–246.

[3] **Fraenkel A.S.**, All about the Responsa Retrieval Project you always wanted to know but were afraid to ask, Expanded Summary, *Jurimetrics J.* **16** (1976) 149–156.

[4] **Fraenkel A.S., Klein S.T.**, Novel Compression of sparse Bit-Strings — Preliminary Report, *Combinatorial Algorithms on Words*, NATO ASI Series Vol **F12**, Springer Verlag, Berlin (1985) 169–183.

[5] **Jakobsson M.**, Huffman coding in Bit-Vector Compression, *Inf. Proc. Letters* **7** (1978) 304–307.

[6] **Jakobsson M.**, Evaluation of a Hierarchical Bit-Vector Compression Technique, *Inf. Proc. Letters* **14** (1982) 147–149.

[7] **Knuth D.E.**, *The Art of Computer Programming, Vol.* **II**, *Semi-numerical Algorithms*, Addison-Wesley, Reading, Mass. (1973).

[8] **Nevalainen O., Jakobsson M., Berg R.**, Compression of clustered inverted files, in Proc. of the 7-th Symp. on Math. Foundations of Comp. Sc., Zakopane, Poland (1978) 393–402.

[9] **Schuegraf E.J.**, Compression of large inverted files with hyperbolic term distribution, *Inf. Proc. and Management* **12** (1976) 377–384.

[10] **Teuhola J.**, A Compression method for Clustered Bit-Vectors, *Inf. Proc. Letters* **7** (1978) 308–311.

[11] **Vallarino O.**, On the use of bit-maps for multiple key retrieval, *SIGPLAN Notices, Special Issue* Vol. **II** (1976) 108–114.

[12] **Wedekind H., Härder T.**, *Datenbanksysteme II*, B.-I. Wissenschaftsverlag, Mannheim (1976).

TEXT COMPRESSION USING PREDICTION

Jukka Teuhola and Timo Raita
University Of Turku
Finland

Abstract: In the compression of the text files, the dependencies between the successive characters should be exploited to as great an extent as possible. There are two obvious possibilities: either to detect and encode often occurring character strings, or to encode successors of character blocks. This paper presents two methods based on the latter approach. In the first method we encode only the most probable successors of blocks, whereas in the second we encode them all, using the knowledge of their distribution. The second method uses recursion to store effectively the dependencies between the characters and this results in good compression gains in practical cases.

1. Introduction

Several text compression algorithms have been presented [CLAR72], [MAYN74], [MCCA73], [RUBI76], [SCHU73], which use some kind of heuristics to find "optimal" frequently occurring substrings from the text to be substituted by shorter codewords. Some methods use also a fixed code table to squeeze the original data into more compact form. We discuss here a form of statistical encoding which is done in the spirit of prediction [SHAN51], and which seems to lead to good compression gains.

In the next section we shall study the basic method which has been used in our compression algorithms and the earlier work done on the subject. In section 3 we present some observations of the dependencies between characters in typical text files. On the basis of these results we implemented two efficient text compression algorithms which are described in section 4. In section 5 the efficiency of the algorithms is compared to other well-known algorithms.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

2. How and to what extent to predict ?

The principle of prediction is simple: given a piece of text, say $s_n s_{n+1} \ldots s_{n+k-1}$, where each s_i represents a character, the problem is to guess what the following character s_{n+k} is. This responds to the idea of treating the string as an outcome from a k`th order Markov source, where the probability of occurrence of the next character is dependent on the k previously occurring characters. According to Guiasu [GUIA77], the intersymbol influence extends even to 30 previous letters in a typical English text. The dependencies are strongest to the 1 - 5 previous letters and weaken noticeably at word boundaries.

The predictive encoding technique has been studied in text compression earlier. Ott [OTT67] has presented the probabilistic approach to prediction and its use in data compression. On the grounds of Ott`s paper, Mommens and Raviv [MOMM74] have used a first order Markov process as their model to generate different Huffman codes for the successors of each character. Thus they have managed to construct a better coding scheme than the ordinary Huffman coding where the correlations are disregarded.

The predictive encoding is an ideal way to remove the redundancy from the text when identical predictors are available in encoding and decoding. To guarantee that the predictors always make the right guess, we must have exact knowledge about the nature of the text. This implies that we have statistics gathered from the text and also that the piece of text we are using to predict the next character is long enough to make the guess a unique one.

These assumptions make this approach to data compression somewhat unpleasant because the statistics must be recorded into a code table which is used to expand the coded file into its original form. If we think e.g. a piece of text written in English, usually several previous characters are needed to uniquely determine the next character. The case becomes even worse when the same words, phrases or sentences appear more than once in the text. This means that the phrases and sentences along with some preceding characters must be included in the code table as

such. Even though the strings were stored efficiently using advanced data structures, the code table would grow into untolerable size for the compression technique to be practical. In order to avoid these difficulties, some other ways must be used to determine those characters which are not unique if only a limited amount of previous characters were allowed to be used in prediction.

It must be noted that if the prediction rules are efficiently included in the code table, no other knowledge about the original file, excluding perhaps a few first characters, need to be retained in the coded form of the file.

There are two approaches we can take to make the size of the code table reasonable:

a) we can always use a fixed length block of previous characters, called the prediction block, in guessing the next character

b) the prediction block is chosen to be of a variable length, depending on how many preceding characters are needed to uniquely determine the next character. For practical reasons, the maximal length of the prediction block must be limited to some predetermined value.

In both cases, if we fail to define the character uniquely, it must be stored as such or using some coding scheme.

3. Statistical analysis of text files

In order to see how far the dependencies between the characters extend and how strong they are, some experiments were made. For the research object we selected randomly files, which contained Pascal source programs as well as English text. From the contents of a file we formed a trie structure illustrated in Figure 1.

The height n of the tree is determined by the length of the character block (n - 1), the successors of which we want to examine. On the highest level all characters which appear in the file (possibly excluding the n - 1 last ones) are linked together. The link son(i) of the node i leads on the next level to a list which contains all the successors of the block containing the characters which are on the path from the root to the node i. The lists which are on the lowest level are sorted in descending order by the frequency field. The tree has the following interesting properties:

a) the number of the most probable character occurrences in the original file following a block of length n - 1 (denoted in the following as P(n-1)) can be found by summing the frequencies of the nodes which start the lowest level lists (in the Figure 1.c. P(2)= 8),

b) the number of the unique character occurrences in the original file following the block of length n - 1 (denoted as U(n-1)) is found by summing only those frequencies of a), where the brother link of the first node in the lowest level list is nil (in the Figure 1.c. U(2)=6),

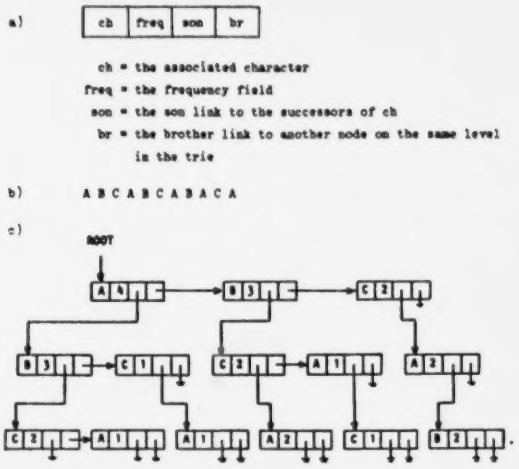

a)
ch
freq
son
br
ch = the associated character
freq = the frequency field
son = the son link to the successors of ch
br = the brother link to another node on the same level in the trie
b)
A B C A B C A B A C A
c)
ROOT

Figure 1. a) The structure of a trie node
b) The contents of an example file
c) The tree of height 3 corresponding to the example text file. The length of prediction block is 2.

c) if there is a node j on level k ($k \geq 2$) with freq(j) = freq(m) and ch(j) = ch(m) for some node m on the first level, then the character is always preceded by a block of length k - 1, which contains the characters along the branch from top level to level k - 1 (from the Figure 1.c. we can see that B is always preceded by A),

d) the number of different blocks of length n can be calculated by summing up the number of leaves,

e) freq(i) = Σ freq(j) for all nodes i, where the summation is done over all leaves j of the i-rooted subtree.

In the following some statistics are shown which illustrate the different aspects of the inspected files. It must be noted that all calculations here and after are based on the 7-bit ASCII code.

The same set of files was used in all subsequent experiments: 5 Pascal source files and 5 English text files, having the lengths of 5 000, 10 000, 15 000, 20 000 and 25 000 characters. Table 1 shows the number of different n-grams (n = 1, 2, 3, 4, 5, 6).

Table 1. The number of different blocks in the inspected files

	The number of blocks of length n in					
	Pascal source files			English text		
n	Min	Max	Average	Min	Max	Average
1	73	89	80	66	82	75
2	557	1065	821	550	931	767
3	1118	2806	2025	1649	3201	2582
4	1527	4477	3099	2669	6674	4974
5	1889	5977	4053	3304	10121	7097
6	2249	7521	5009	3724	14045	9008

The table shows that the files have contained almost all but control characters from the ASCII code set. Although the text files have contained less characters than the program source files on the average, the diversity of the blocks is greater because of the less regularity in expressions. The table also shows how the trie is expected to grow as n is increased.

In Table 2, the number of unique and most often occurring characters following a block of length n (n = 1, 2, 3, 4, 5) are shown. The table exhibits the possibly surprising fact, that one preceding character is sufficient to predict correctly the next character in approximately 36% of all the cases (the average of the P(1) values). If the length of the prediction block is increased from one to three characters, the percentage of correct prediction still grows noticeably. The advantage gained with still larger blocks becomes less.

Table 2. The percentual share of the most probable and unique characters following a prediction block of length i.

Length of prediction block, i	Pascal source files		English text	
	P(i)	U(i)	P(i)	U(i)
1	44.1	4.8	27.8	1.2
2	69.3	24.8	47.2	6.7
3	80.9	48.9	67.5	27.4
4	84.5	59.2	79.2	52.9
5	86.7	64.9	85.5	68.4

On the basis of Table 2 it is evident that the lengths of the chains on the lower levels are shorter the deeper we go into the tree. Table 3 summarizes the average fan-out of nodes on each level of the tree.

Table 3. The average fan-outs of trie nodes on different levels.

Level number	Pascal source files	English text
1	80.4	75.4
2	10.2	10.2
3	2.5	3.4
4	1.5	1.9
5	1.3	1.4
6	1.2	1.2

4. Two efficient compression methods based on the prediction principle

According to Table 2, the majority of the characters can be defined uniquely by using only a few of the previous characters. On the other hand, the size of the code table grows quickly and therefore prevents the use of much longer text fragments. These facts in mind we implemented two different versions of the prediction method.

Version 1. The file to be compressed is scanned for all n-grams (n fixed) it includes. The n-grams are partitioned into subsets so that within each subset all n-grams have common first n - 1 characters. The subsets are sorted by frequency. For each subset the most often occurring n-gram is selected for further use and the others are discarded. This results in a set of n-grams $S = \{s_{i1}s_{i2}\ldots s_{in} \mid i = 1,\ldots,m\}$ where no $s_{i1}\ldots s_{i,n-1} = s_{j1}\ldots s_{j,n-1}$ $(i \neq j)$. The original file is then encoded in another pass as follows: the very first n - 1 characters of the file are stored as such. Let us suppose that the encoding process has proceeded to the k`th character ($k \geq n$) of the original file. If $s_{k-n+1}s_{k-n+2}\ldots s_k$ is in the set S, we can move to examine character $s_{k+1}$. Otherwise, $s_k$ and its index k` are stored to the set of leftover characters and the set of indexes respectively. This is done until the original file is searched through.

The indexes of the leftover characters could be stored as such or as the distances of successive indexes. We chose a third alternative, a bitmap, the size of which could be reduced by using an appropriate bit string compression method.

In order to see the relative share of different data sets, a table which shows the relative sizes of the trie, the leftover character set and the bitmap was formed (Table 4.).

Table 4. The average sizes the different data sets in version 1.

The length of the prediction block	Pascal source files: Average percentual size of			English text: Average percentual size of		
	The trie	The bitmap	Leftover char set	The trie	The bitmap	Leftover char set
1	2.0	19.6	78.4	1.5	15.9	82.6
2	24.1	23.7	52.2	16.9	17.4	65.7
3	56.6	18.2	25.2	52.3	14.2	33.5
4	74.5	12.0	13.5	77.5	8.9	13.6
5	84.0	8.1	7.9	89.4	5.2	5.4

As was expected, the size of the trie grows quickly dominant with respect to the amount of the other data. The sum of the trie size and the leftover characters reaches its minimum with the prediction block length of 2. The advantage which is gained with larger block lengths is overweighted by the quickly growing trie size. If the trie structure and the trie characters are coded efficiently, i.e. using some other code than ASCII, it might be possible to store the trie into a more compact form. Consequently, the relative importance of the data sets might change. The distribution of ones and zeroes in the bitmap prevents us from gaining any additional savings by using a bit string compression method. Therefore, version 1 was implemented by storing the bitmap to the result file as such.

As the result of the coding, the compressed file consists of three components: the trie, the leftover characters and the bitmap. The trie is stored by starting at the top level and recording the lengths of the lists at each level. At the same time the characters are gathered from the nodes. As a final procedure the node characters and the leftover characters are collected together and Huffman-coded. The results of the version are shown in Table 5.

Table 1. The performance of the prediction method (version 1) for various prediction block lengths.

File type	File length (characters)	The compression gain when using the prediction block length of 1	2	3	4	5
Pascal source programs	5 000	37.2	39.4	13.0	-32.0	-86.0
	10 000	37.9	43.7	23.0	-18.7	-71.0
	15 000	40.7	50.5	39.2	10.4	-27.5
	20 000	38.8	53.2	43.1	17.3	-17.3
	25 000	40.8	52.1	42.7	17.3	-16.7
Average	15 000	39.1	47.3	32.2	-1.1	-43.7
English text	5 000	31.0	29.0	-3.0	-71.9	-160.4
	10 000	34.4	36.4	18.0	-32.1	-104.2
	15 000	35.7	39.7	26.5	-14.8	-78.3
	20 000	35.5	40.5	29.6	-8.8	-69.0
	25 000	36.7	42.8	36.6	5.5	-64.8
Average	15 000	34.7	37.7	21.5	-24.4	-95.3

The test results show that a short prediction block is preferable for short files and a longer block for longer files. This phenomenon is due to the fact that although the storage needed for the trie grows with longer prediction block, the knowledge about the nature of the original character string contained in the trie is sufficient to overcome this growth with large files. This suggests that the performance of the prediction method becomes the better the longer the original file is. It is also expected that with files of 50 000 - 200 000 characters, the method performs better with block size 3 than with sizes 1 or 2.
The optimum prediction block size from small files is one and for files of length 10 000 - 20 000 characters two. The best compression results were all obtained with these values.

It is evident that version 1 can be easily improved if the trie is stored efficiently. Therefore, version 2 was implemented.

Version 2. In version 1 only the most probable successors of blocks were considered. The number of successor characters per prediction block is usually quite small and it decreases when the block length is increased. If we store all successors of blocks as leaves in the prediction tree, we can encode all but n - 1 first characters of the file by small integers expressing the position of each character among its brother nodes on the leaf level (which is sorted by frequency within each parent). The sequence of small integers is encoded using e.g. the Huffman method. Even though the distribution of these integers varies for different blocks, we can content ourselves to a single Huffman code table since the probability distribution of integers in each set of successors is rather similar. For example, in the case of Fig. 1.b. for block length 2, the sequence of integers is

111111211

preceded by the first two characters "AB". If some leaf of the prediction tree has no brothers, the leaf character is a unique successor of the block represented by the path to it in the trie. The sequence number 1 of the unique successor need not be represented at all in the compressed file. Applying this technique we can compact the sequence of integers in the example case further to

112.

The size of the prediction tree is in this case considerably greater than in version 1 (for the same block length). In trying to compact the tree we ended up to the following solution, which in fact utilizes the described coding recursively with reduced block length:

- characters on level 1 are stored as such (ASCII codes)
- assume that there is a path $c_1,\ldots,c_k$ from the root to level k. When storing the tree, node c_k can be encoded with the number which expresses its sequence number among the brothers at the end of path $c_2,\ldots,c_k$ from the root. (On this path each c_i resides on level i - 1.) Now we must assume that the brothers are sorted within their parents by frequency on all levels. It is obvious that the required path is present in the tree except possibly for n - 1 last characters of the file. In order to guarantee correct operation also for them we extend the original file by n - 1 spaces, which are not included in encoding.

Figure 2 shows the same tree as in Fig. 1.c., equipped with code numbers, of which the unique ones are enclosed in brackets. E.g. the 5th leaf node C is encoded with the number 2, because its parent is A and on level 2 C is the second among A's children.

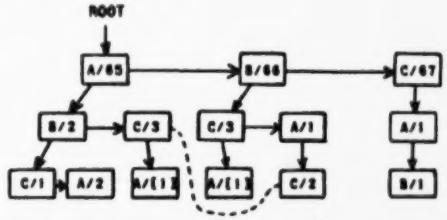


Figure 2. The sample prediction tree with attached code numbers (the first level nodes having ASCII codes). A sample correlation is indicated: C on the path B-A-C is encoded by 2, because C is the second of A:s children on level 2. The codes in []-brackets may be omitted.

In this way we can replace the tree nodes by small integers, which tend to be smaller, the lower we are in the tree (because the fan-outs of nodes usually decrease when we proceed downwards). It is thus sensible to further encode each level separately using the Huffman method. Note that also here the unique 1-codes can be discarded.

In the decoding phase the tree is rebuilt in an obvious fashion: level by level, and using the sequence numbers we pick up the characters from the previous level.

Table 6 illustrates the proportions of different data sets within the compressed files.

Table 6. The average sizes of different data sets in version 2.

The length of the prediction block	Pascal source files: Average percentual size of			English text: Average percentual size of		
	Shape of the tree	Nodes of the tree	Coded file	Shape of the tree	Nodes of the tree	Coded file
1	1.5	9.8	88.8	1.7	11.7	86.6
2	5.3	29.1	65.5	6.7	36.1	57.2
3	12.2	51.0	36.8	12.9	52.3	34.8
4	19.1	61.3	19.6	18.6	56.6	24.8
5	25.3	63.2	11.4	24.3	57.8	17.9

The proportion of the shape of the tree grows rather steadily. Similarly, the proportion of the actual coded file decreases smoothly. Instead, the storage of the tree nodes soon becomes dominating (for blocks ≥ 3), but the growth of the proportion slows down, to a level around 60% of the encoded file. This property is due to the feature of using the prediction also when storing the tree nodes.

Table 7 shows the compression gains for all ten test files and for prediction block lengths from 1 to 5.

Table 7. The performance of the prediction method (version 2) for various prediction block lengths.

File type	File length (characters)	The compression gain when using the prediction block length of				
		1	2	3	4	5
Pascal source programs	5 000	37.7	43.3	37.7	28.3	18.3
	10 000	43.2	49.2	43.1	34.0	24.9
	15 000	47.2	52.5	47.3	40.7	32.9
	20 000	46.8	50.4	48.8	43.0	36.2
	25 000	48.3	60.0	58.2	52.2	45.4
Average	15 000	44.6	54.3	51.0	43.6	35.5
English text	5 000	31.6	24.8	11.9	-4.4	-20.0
	10 000	38.2	36.6	27.3	13.2	-1.9
	15 000	41.4	41.6	34.9	21.5	6.9
	20 000	41.5	43.1	37.4	25.0	10.7
	25 000	43.6	46.7	43.3	32.8	19.7
Average	15 000	39.3	38.6	31.0	17.6	3.1

We see that version 2 is superior to version 1 in all cases. The gains for larger Pascal files are around 60%, which can be regarded as a very good value. No information-theoretic lower bound can be given for these kinds of files, because the dependencies between text portions are very intricate.

5. Conclusions

Two methods were presented for compressing text files, both of which are based on the idea of prediction, where we take advantage of the local dependencies occurring in the text. The distribution of the successors of a given block of characters is usually extremely skew, facilitating compact encoding of the successors. There is of course a trade-off between the prediction power and the size of the index which stores the successors of blocks. A prediction block of length 2 was observed to give the best compression gains for practical file sizes.

Let us finally compare the above results with two other well-known methods: the Huffman coding of characters and the so called incremental coding presented by Rubin [RUBI76] (in fact a faster modification of it [RAIT85]). Table 8 shows the average compression gains achieved by the four methods. In prediction coding the average is calculated from the best prediction block length for each test file.

Table 8. The percentual compression gains achieved with prediction, incremental and Huffman coding. Note that the calculations are based on the 7-bit ASCII code and that all data which is needed to decompress the file is stored within the compressed form.

File type	Prediction coding		Incremental coding	Huffman coding
	Version 1	Version 2		
Pascal source files	47.8	54.3	39.4	27.0
English text	38.1	40.2	32.9	34.2

Version 2 is clearly the best of the four methods and version 1 the second best. The test cases were somewhat unfair for incremental coding, because there were not enough unused characters, which the method can take advantage of. In addition, although the distribution of distinct characters is skew, the distribution of n-grams ($n = 2, 3, \ldots$) isn't, which can also be seen from the Tables 1 and 3. This favors Huffman coding to the extent that it performs even better than the incremental coding. By applying the Huffman coding to the result of incremental coding the result could be improved - according to our tests the combined compression gain would be 43.0% for Pascal files and 39.1% for English text files.

The method in the form presented above works quite slowly. The average encoding time was 1.25 seconds/Kbyte for block length 2 on a DEC-2060 computer. The algorithm was implemented in Pascal and no optimizing was made either in the program code or in the compilation. The speed can be improved by replacing the Huffman coding phase by some faster non-optimal variable-length coding scheme. In fact, we implemented a "production version" by applying the universal coding presented in [TEUH78]. The compression gains did not reduce, instead, the average encoding time decreased to 0.54 seconds/Kbyte. The decoding time is about half of this.

References

[CLAR72] Clare, A.C., Cook, E.M. & Lynch, M.F.: The identification of variable-length, equifrequent character strings in a natural language data base, The Computer Journal, Vol 15, No 3, 1972, pp. 259-262

[GUIA77] Guiasu, S.: Information theory with applications, McGraw-Hill, London, 1977

[MAYN74] Mayne, A. & James, E.B.: Information

compression by factorising common strings,
The Computer Journal, Vol 18, No 2, 1974, pp. 157-160

[MCCA73] McCarthy, J.P.: Automatic file compression,
International Computing Symposium 1973, North-Holland, 1974, pp. 511-516

[MOMM74] Mommens, J.H. & Raviv, J.: Coding for data compaction,
IBM Research Report RC5150, T.J. Watson Reasearch Center, Yorktown Heights, NY, 1974

[OTT67] Ott, E.: Compact encoding of stationary Markov sources,
IEEE Trans. On Inf. Theory, Vol IT-1, No 1, January 1967, pp. 82-86

[RAIT85] Raita, T.: An automatic system for file compression,
To be published in The Computer Journal

[RUBI76] Rubin, F.: Experiments in text file compression,
Comm. Of The ACM, Vol 19, No 11, November 1976, pp. 617-623

[SCHU73] Schuegraf, E.J. & Heaps, H.S.: Selection of equifrequent word fragments for information retrieval,
Inf. Stor. Retr., Vol 9, 1973, pp. 697-711

[SHAN51] Shannon, C.E.: Prediction and entropy of printed English,
Bell System Tech. Journal, 30, 1951, pp. 50-64

[TEUH78] Teuhola, J.: A compression method for clustered bit-vectors,
Inf. Proc. Letters, Vol 7, No 6, 1978, pp. 308-311

Incorporating Syntactic Information into a Document Retrieval Strategy: an Investigation

Alan F. Smeaton
Department of Computer Science
University College Dublin
Belfield, Dublin 4
Ireland.

Abstract

This paper deals with mechanisms for performing text retrieval which incorporate a degree of linguistic processing into the overall strategy. We have performed some experiments using parsing of text an a test collection of documents and queries to try and find out exactly if and how parsing could contribute to an overall improvement in retrieval effectiveness. Investigating this topic has led us to the definition of a retrieval strategy which incorporates parsing of query text and a more "shallow" parsing of document texts, whose retrieval effectiveness is investigated and described. Our results indicate that significant improvements in retrieval effectiveness can be obtained by incorporating such linguistic processing into an overall retrieval strategy.

1 Introduction

This paper deals with mechanisms for performing text retrieval of office objects in an office environment. In particular we concentrate on retrieval of objects where the user specifies some properties of the content of the textual parts of office objects. To date, the most popular retrieval strategies for text objects among researchers have been Boolean retrieval and probabilistically based statistical methods. Both have their disadvantages and are well-documented in [Salton and McGill, 83]. Among the problems with these retrieval strategies is the poor level of retrieval effectiveness.

One possible way to improve on retrieval effectiveness would be to introduce some form of linguistic processing into the overall retrieval mechanism, which would perform some deeper "understanding" of either user queries or the text part of documents, or both, which in turn could contribute to more effective retrieval. The two forms of linguistic processes most frequently suggested for this purpose are semantic and syntactic processing. The problem with using semantic processing on text as part of retrieval is that it is usually domain dependent which means a lot of information on the domain being used must be provided for the system. This restriction on the use of semantics makes semantic processing unsuitable for dynamic environments with a broad coverage as is found in an office environment. The main problem with using parsing or syntactic processing of text as part of retrieval is that it is not known exactly how the information that parsing provides, can be effectively incorporated into a retrieval strategy.

In this paper we will perform some experiments using parsing of text and a test collection of documents and queries to try and find out exactly if and how parsing could contribute to an improvement in retrieval effectiveness. Investigating this topic has led us to the definition of a retrieval strategy incorporating parsing of query text and a more "shallow" parsing of document texts, whose retrieval effectiveness is investigated and described. Before describing the experiments we have performed, however, we will outline some background to our work.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.
© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

2 Background

The understanding of written or spoken natural language is a process usually broken up into several layers or levels. In most natural language processing (NLP) systems, each level works on its own and only communicates with the layers directly above and beneath it. For example Winograd [1983, pp17] proposes a model for language comprehension with the following levels;

- phonological — deals with phonemes - spoken text only
- morphological — deals with individual word forms
- lexical — deals with whole words
- syntactic — deals with groups of words like phrases
- semantic — deals with meanings of words and phrases
- pragmatic

According to Kraft [1985], processing at the morphological level in present information retrieval systems removes prefixes and suffices and reduces words to a single root form. At the lexical level in present information retrieval systems stopwords or non content-bearing words can be identified, while at the syntactic level "... correct phrases for index terms may eventually enable indexing of document subcomponents". The subject area dealt with in this paper is an analysis of how processing text at the syntactic level may be used to improve retrieval effectiveness. Specifically what we want to do is investigate if and how parsing can be used as an aid to the retrieval of natural language text.

An overview of what parsing is, can be found in [Hayes and Carbonell, 1983] which gives a tutorial on NLP applications and techniques, including parsing, and this paper describes such topics as parse trees, context-free, transformational and augmented transition network grammars. Another good introduction to parsing can be found in [Grishman, 1984]. The reader is referred to these papers for further reading on this subject.

The advantages of using syntactic analysis of text (parsing) as part of a retrieval strategy are that it is a general-purpose linguistic technique and it is available. Because parsing is general-purpose it requires no tailoring to specific domains which makes it suitable for an office environment. Recent work [Marsh and Friedman, 1985] however, has suggested that particular application domains may have particular idiosyncrasies with respect to grammar usage, which they calf sub-grammars. In their paper, Marsh and Friedman cite some examples of this, but it seems that the differences between natural language usage in different domains is so minimal as to be negligible. Because parsing is such an integral and important part of any NLP system, it is a well-researched area in the field of artificial intelligence. Much work has been done in the areas of parsing strategies and methods, grammars and types of grammar for natural language, methods of resolving ambiguities, etc. As a consequence of all this work there are off-the-shelf parsers available so that the technique of parsing can be used for non NLP applications as we want to investigate.

Syntactic analysis of text does have its problems which make it a computationally non trivial problem. One of those problems is caused by the ambiguity of a lot of English words (called polysemy) and one of the consequences of word ambiguity is that it can lead to many alternate parse trees for one sentence. All these parses will be syntactically correct but some will be semantic nonsense. In these cases it is the semantics of the words which would be used to disambiguate the parse trees, but there are cases where sentences are semantically and syntactically ambiguous such as the "garden path" sentences ("Put the block in the box on the table in the kitchen"). According to Church and Patil [1982] the sentence

> "In as much as allocating costs is a tough job I would like to have the total costs related to each product"

can have 958 syntactically correct parses using their particular grammar and lexicon !

A second reason why computational parsing can be difficult stems from the fact that there are so many ways to say the one thing in English. Winograd lists 10 ways of saying ...

> "My husband embroidered this jacket for my birthday"

using different syntactic constructs [Winograd 83, pp277], but all with the same semantic content. It is not the job of a parser to resolve all possible ways of saying the same thing into one interpretation; in a NLP system this would be done by some later semantic processing. What does cause problems for the parser is that a grammar for the language most be able to recognise all syntactically correct ways of phrasing something, and this makes grammars very complex and very big. It is this aspect which complicates the parsing mechanism.

Perhaps the greatest difficulties of the English language for understanding purposes is that it is full of phrasal constructs. The word is not always the unit of meaning in English. English, and other natural languages, are full of phrases and some of these have been classified by Wilensky and Arenas [1980] into the following (incomplete) set of phrasal language constructs

- Formulaic Utterances — "How do you do"
- Lexical Collocations — "The Big Apple" (New York)
- Canned Phrases — "a case in point"
- Situation Utterances — "it only hurts when I laugh"
- Cliches — "as sly as a fox"
- ...

as well as lots of other idioms or stock phrases where the meaning of the phrase is not determined by the meaning of its constituents ("kick the bucket", "bury the hatchet", "throw the book at", ...). If these phrases were to be parsed with words as units of information they would make semantic nonsense, as well as giving incorrect sentence parses. In order to manage them properly, it is desirable to treat the phrases themselves as syntactic units. Thus they should be built into a grammar for the language, but the amount of, and undefined nature of, such phrases, makes this difficult. One way of getting around this problem is to have a vocabulary of phrases which are treated as single syntactic units during the parsing process, but this approach is not often adopted in parsing, despite the widespread nature of the use of phrases—especially in the office environment.

One further problem with dealing with text stems from using the sentence as the unit of information. When used as part of a larger corpus of text, a sentence will usually be related to either the previous or later sentences, or both. This usually takes the form of the use of pronouns, ellipses, etc. Correct resolution of pronouns is a process requiring semantic processing, and is not part of parsing which would process sentences one at a time. All these problems inherent in any parsing of natural language, and others, have been identified as long as twenty years ago in [Salton, 1971, pp116–117].

The problems of ambiguity in noun phrases and prepositional phrases only has been highlighted by Salton [85] who gives examples of multiple interpretations of some two-term noun phrases. In this paper Salton proposes the use of case grammars which include a small amount of supposedly domain-independent semantic information which he believes will resolve some ambiguity. While the ambiguity resolution part may be true, the price to be paid for such hoped-for improvement in retrieval effectiveness is a much more complex linguistic processing component. In contrast to this approach what we do in this paper is concentrate on a less complex linguistic processing technique and exploit its capabilities to a fuller extent. To be precise, we have carried out a series of experiments on a test collection of documents and queries which investigate how parsing of natural language text can be used as part of a retrieval mechanism. We have done this by performing a statistically-based retrieval to give us some benchmark results and then we have analysed these benchmark results using syntactic information, to see if and how syntactic information could be used to further improve retrieval effectiveness. From the analysis of our early results, a retrieval strategy which incorporates some syntactic processing and is integrated with statistically-based retrieval, is suggested and tried.

3 Experimental Setup

The collection of test data used in the experiment consists of the titles, abstracts and bibliographic information from papers appearing in the Communications of the ACM in the period 1958

to 1979, as well as some user queries and user relevance judgments on the collection. This test collection was originally put together by Fox and his co-workers at Cornell University and is described in more detail in [Fox 83]. One of the reasons for using the CACM collection was that the subject matter of documents in the collection, computer science, is a subject familiar to the author of this paper, thus making manual parsing and analysis of queries and documents a more manageable task.

Each of the 3204 documents in the original test collection consists of a title, a full abstract, a list of authors, ACM keyword lists, ACM Computing Review categories, date of CACM publication and some co-citation information. Because the experiments reported in this paper are concerned with retrieval directly from raw text rather than using the more structured parts of documents, only the title and abstract fields were used in these experiments. These text fields were stemmed and had stopwords removed, which gave a set of word stems to be used as index terms for each document as an internal representation for retrieval [Porter 80].

The 64 queries and relevance judgments provided with the CACM collection were originally formulated by computer science department personnel at Cornell University and elsewhere in the U.S. These queries consist of natural language paragraphs stating a specific interest in some topic in computer science. The 64 queries were reduced to 25 for various reasons. For example some queries had no relevant documents at all while others had too many (more than 20), some queries asked for a specific reference and others consisted of more than one distinct information need. Eleven of the reduced set of 25 queries consist of grammatically well-formed text sentences (hereafter called the TextQ query set) while the remaining set of 14 queries consist of a single noun phrase (NP), some conjoined NPs or a simple list of syntactically unlinked NPs (this set of queries to be called the PhraseQ set). The complete set of 25 queries will be called the TotalQ set. The reason for differentiating between query types is to see whether one particular type of query is effected more than another, by the use of syntactically derived information in retrieval. All the queries were stemmed in the same way as the documents which gave a set of query terms for each query.

For each of the 25 queries, the complete document collection was ranked using the well-known IDF ranking method [van Rijsbergen 79]. Here weights were calculated for each query term, given as:

$$w_i = -\log n_i/N$$

where n_i is the number of documents in the collection indexed by term i and N is the total number of documents (3204 in this case). The ranking "score" for each document is given as the sum of the query term weights of the query terms indexing that document.

An arbitrary cutoff point of 20 was chosen for the collection ranking for each query, which gave 3 categories of documents for each query. These categories are:

- REL/RETR—relevant documents retrieved above the cutoff
- NON/RETR—non-relevant documents retrieved above the cutoff
- REL/NON—relevant documents not retrieved above the cutoff

Table 1 gives a breakdown of the number of documents in each category for each query type, summed over all queries.

	TotalQ	PhraseQ	TextQ
REL/RETR	75	45	30
NON/RETR	415	224	191
REL/NON	171	99	72

Table 1: Numbers of Documents in Each Category

One of the 25 queries retrieved ALL of its relevant documents in the top 10 positions in the IDF-ranking which explains why the entry in REL/RETR and NON/RETR for the TotalQ category of queries totals 490 and not 500. Altogether, the 25 queries retrieved 75 (average 3.0) of the 246 (average 9.8) relevant documents in the top 20 rank positions. This leaves 171 (average 6.8) relevant documents not retrieved in this top 20.

In these experiments, the way in which we have used syntactic information is to allow the syntactic structure of phrases and sentences to define dependencies between words or word stems (hereafter referred to as terms), in the users query. We do this processing on user queries only because it is done manually and is thus impossible to do for the entire 3204 document texts. This is just one possible way of using syntactic information but as mentioned earlier, the purpose of these experiments is to determine whether syntactic information could be used in retrieval and not to define the most effective way of incorporating this information into retrieval. Thus using syntactic structures to determine term dependencies is a reasonable starting point for this line of research.

In order to derive sets of dependencies or to mark sets of dependent terms from the syntax of user queries, some heuristic rules were used. These rules are given in Appendix 1. There is no theoretical background to the dērivation of these rules; they are completely arbitrary and could certainly be improved and refined. The justification for using them is that a first approximation of a set of rules will suffice for our purpose. The rules given in Appendix 1 will derive sets of dependent pairs, triples, quadruples, etc., where each element in the dependency is a word stem. Table 2 gives the average number of dependencies derived for each query type.

	TotalQ	PhraseQ	TextQ
Av. No. dependent pairs	6.6	4.7	9.1
Av. No. dependent triples	5.3	3.2	7.8
Av. No. dependent quadruples	2.4	1.2	3.7
Av. No. dependent quintets	0.5	0.1	1.0
Av. No. dependent sextets	0.1	-	0.3

Table 2: Average Numbers of Dependencies

On average, PhraseQ queries contained 8.4 words (2.3 Noun Phrases) while TextQ queries contained 30 words. This would explain why TextQ queries led to more than twice the average number of dependencies than PhraseQ queries. Also, because TextQ queries consisted of grammatically well-formed sentences, they also contained a lot of non-content bearing words and phrases like "Please give me ... ", "have you any information on ... ". This fact may have an effect on results later on.

For each query, what we have done is to manually parse the text of each document in the three categories REL/RETR, NON/RETR and REL/NON, searching for the occurrence of dependent terms derived from the query, within sentences and also within phrases or clauses of the document text. The reason for doing this is that it requires a very "shallow" parsing of document text needing only to identify phrase and sentence boundaries, a process which should be easy to implement au-

tomatically. Certainly the parsing of document texts is far less deep than that carried out on user queries. Also, since document texts are only being parsed so as to identify query term co-occurrance within sentences and within phrases, we need only parse the part of the document texts surrounding the locations of query terms. This makes this process easier still. The results of searching for query term co-occurrance in document texts will indicate whether syntactic information from a user query can be used in conjunction with shallow syntactic information in filed documents, in a retrieval mechanism. The results of this searching are reported in the next section.

4 Experimental Results

For practical purposes we have limited our searching for dependencies from queries to pairs of dependent terms occurring in the same sentence in the document text (SS2) and in the same phrase in the document text (SP2), and to triples of dependent terms in the same sentence (SS3) and in the same phrase (SP3) of the document text. This has been done because it was found that there are few, if any, occurrences of dependent quadruples, quintets, etc. of dependent terms in document texts, although forming such large-numbered dependencies from user queries is useful as it helps to formulate other dependencies. In the context of searching for dependencies in documents, phrases are defined as noun phrases, verb phrases, prepositional phrases, etc., in fact any grouping of words which makes up a logical unit.

Tables 3 to 5 summarise the statistical results found from dependency searching.

These tables show the actual numbers, and averages in parentheses, of occurrences of SS2, SP2, SS3 and SP3 in the three categories of documents, for the three query groups. The occurrence of an SS2 or SS3 means that a pair or triple of dependent terms, syntactically identified as dependent from the query, has occurred in the text of a document, within the same sentence. The occurrence of an SP2 or SP3 means that a pair or triple of dependent terms has occurred in the same phrase, and by implication, in the same sentence, of a document, although this latter statistic is not gathered. Thus the true figures for SS2 and SS3 occurrences are those given above, added to the SP2 and SP3 figures respectively.

	REL/RETR	NON/RETR	REL/NON
SP2	58 (0.77)	102 (0.25)	75 (0.44)
SS2	23 (0.31)	107 (0.26)	37 (0.22)
SP3	18 (0.24)	29 (0.07)	8 (0.05)
SS3	20 (0.27)	13 (0.05)	11 (0.06)
Docs in categories	75	415	171

Table 3: No. dependency occurrances (and Averages), TotalQ Queries

	REL/RETR	NON/RETR	REL/NON
SP2	31 (0.69)	43 (0.19)	26 (0.26)
SS2	13 (0.27)	60 (0.27)	20 (0.20)
SP3	7 (0.16)	11 (0.05)	2 (0.02)
SS3	10 (0.27)	8 (0.04)	3 (0.03)
Docs in categories	45	224	99

Table 4: No. dependency occurrances (and Averages), PhraseQ Queries

	REL/RETR	NON/RETR	REL/NON
SP2	27 (0.9)	69 (0.36)	49 (0.68)
SS2	10 (0.33)	47 (0.24)	17 (0.24)
SP3	11 (0.36)	18 (0.09)	6 (0.08)
SS3	10 (0.33)	5 (0.3)	8 (0.11)
Docs in categories	30	191	72

Table 5: No. dependency occurrances (and Averages), TextQ Queries

One fact to emerge from the compilation of the above statistics is that there are many relevant documents which contain none of the original query terms for the query in question. This means that such relevant documents are effectively irretrievable without some kind of query expansion process. Table 6 shows exactly how many relevant and "irretrievable" documents (REL/IRR) there are in the CACM collection.

	REL/NON	REL/IRR
TotalQ	171	30
PhraseQ	99	26
TextQ	72	4

Table 6: No. Relevant and "Irretrievable" Docs.

The huge difference in the number of "irretrievable" relevant documents between the two query types can be attributed to the fact that TextQ queries have, on average, more words in the user query (30 as against 8.4 for PhraseQ queries) and hence they will have more query terms which can occur in relevant documents. The fact that 30 out of 171 REL/NON documents are irretrievable without query expansion may be useful in later analysis of results.

The statistics in Tables 3 to 5 lead to a number of observations. Firstly the column of averages for REL/NON is higher than that for NON/RETR, in particular for SP2 entries for all query categories while SS2 and SS3 entries are about equal and SP3 entries show higher averages for NON/RETR than for REL/NON. As noted earlier though, the true SS2 and SS3 averages are the SS2 and SS3 entries plus the SP2 and SP3 entries respectively, as SPn entries imply SSn entries which are not included in the statistics. Taking this fact into account, SP2, SS2 and SS3 averages are higher for REL/NON than for NON/RETR. What this result suggests at this point is that SP2, SS2 and SS3 characteristics are potentially useful as an aid to retrieving some REL/NON documents before some NON/RETR, thus leading to an improvement in retrieval effectiveness, though this must be investigated further.

A second observation to be made about the results so far is that almost all the averages in Table 5 (TextQ queries) are higher than those in 4 (PhraseQ queries), with averages in Table 3 (TotalQ queries) in between the two. This is not unexpected since TextQ queries give twice the amount of dependent pairs, triples, etc. as PhraseQ queries, with TotalQ queries, the union of the two other sets of queries, in the middle.

The actual numbers of occurrences and averages, in Tables 3 to 5, are quite small. For example all averages are below 1.0. This statistic alone is difficult to interpret without knowing about the distribution of SSn's and SPn's among the documents in each category. In order to get a better feel of what these distributions are, of SSn's and SPn's are made up, for the TotalQ query set.

What the results in table 7 shows is that the dependency occurrences are distributed in such a way that there are some exceptional individual documents which have as many as FOUR SP3 occurrences in one document (REL/RETR), or others which have SIX SP2 occurrences in one document (REL/NON, NON/RETR. As a consequence of this there are also some relevant documents,

Document Category: REL/RETR (75 docs)

No occurrences:	1	2	3	4	5
SP2	30	9	2	1	-
SS2	17	3	-	-	-
SP3	7	2	1	1	-
SS3	16	2	-	-	-

Docs with NO dependency occurrences: 18

Document Category: NON/RETR (415 docs)

No occurrences:	1	2	3	4	5	6
SP2	46	14	1	3	1	3
SS2	59	16	-	1	-	2
SP3	16	3	1	1	-	-
SS3	13	-	-	-	-	-

Docs with NO dependency occurrences: 248

Document Category: REL/NON (171 docs)

Occurrences:	1	2	3	4	5	6
SP2	30	11	3	2	-	1
SS2	21	5	-	-	-	1
SP3	6	1	-	-	-	-
SS3	9	1	-	-	-	-

Docs with NO dependency occurrences: 106

Table 7: No. Dependency Occurrances, TotalQ set

both in the REL/RETR and REL/NON categories, which have no occurrence of syntactic dependency in the document texts. For example, 18 of the 75 relevant and retrieved documents have NO syntactic dependency occurrences which makes one wonder what makes them relevant ?

The fact that 248 of the 415 documents in the NON/RETR category have no syntactic dependency occurrence in their document texts suggests the retrieval of some of the documents in this category could be further "demoted" in the document ranking if syntactic dependency occurrence was used as part of a document ranking function. If such a factor is used as part of a retrieval function then 106 of the 171 documents in the REL/NON category would be "demoted" in the document ranking, as would 18 of the 75 documents in the REL/RETR with no syntactic dependency occurrence! However it is possible that some of the 65 documents in the REL/NON category which DO have syntactic dependency occurrence in document texts, may be promoted in their place to a sufficient extent so as to lead to an overall improvement in retrieval effectiveness.

In a recent paper by Croft [85] experiments are reported using the same test collection of documents and queries as we have used. In one of the reported experiments the co-occurrence of query terms extracted from phrases specifically identified by users as important, in document texts, led to those document scores being incremented by some factor. Thus if a document is indexed by 2 query terms which also co-occur in a phrase identified by a user as important, the score for that document, which is used to determine collection ranking, is incremented. The incrementing of document scores in this way is based on a probabilistic model of retrieval described in Croft's paper. In the experiments reported in this paper we have also incremented document scores by some factor but the differences between our experiment and Croft's work are the methods used for identifying dependencies from queries, and the fact that syntactic information from document texts (sentence and phrase boundaries) is used in our work.

The method used for evaluating retrieval effectiveness in these experiments is to calculate recall-precision values at standard recall levels of 10,20,... 100 which is described in more detail in [van Rijsbergen, 79]. Table 8 shows precision values for the IDF ranking method and also for the "corrected" IDF method where document scores have been incremented by different values depending on the occurrence of dependent pairs and triples in sentences and phrases of document texts.

What happens in the corrected strategies is that a document score is calculated as the sum of all the query term weights indexing that document, plus the corrective weights for any pairs or triples of dependent terms occurring in the same sentence or phrase in the document texts. To put the values of the corrective weights into perspective, the IDF weight given to a query term occurring in 674 of the 3204 documents in the collection is 6, while the weight given to a term occurring in 4 documents is 29. (Real IDF weights have been multiplied by a constant and rounded to make them integers.) The corrective weight has been added to the top 50 IDF-ranked documents only as documents outside this top 50 are unlikely to have many co-occurrences of query terms.

IDF Uncorrected	Corrected 5,10,10,15	Corrected 5,10,5,10	Corrected 5,15,5,15
55.5	62.5	61.6	60.3
37.4	47.1	46.0	46.7
27.1	35.5	35.9	36.4
26.9	34.0	35.7	33.5
21.1	26.5	28.2	25.8
16.0	16.7	18.0	15.2
6.4	8.5	8.5	8.5
5.9	8.0	8.0	8.0
3.8	6.0	6.0	6.0
2.8	3.5	3.5	3.5

Table 8: Precision values where figures in heading refer to weights given to the occurrence of SS2,SP2,SS3 and SP3 dependencies respectively

As can be seen from Table 8, the correction factor leads to a quite significant improvement in retrieval effectiveness. Of the various combinations of corrective weights the 5/10/10/15 permutation gives most improvement. To compare this improvement with other work, we have extracted the same 25 queries as we have used from the results obtained by Croft [85] and evaluated them in the same way as our own results. In Croft's experiments he has used a different document indexing strategy and so has obtained a different set of benchmark IDF results. His results are given in Table 9. and Croft's results, together with our benchmark IDF and our best corrected strategy results are given in Figure 1

As can be seen from the graph in Fig 1 the relative improvement obtained by using our corrected strategy over our benchmark, is greater than the relative improvement obtained by Croft in his corrected strategy over his benchmark. These

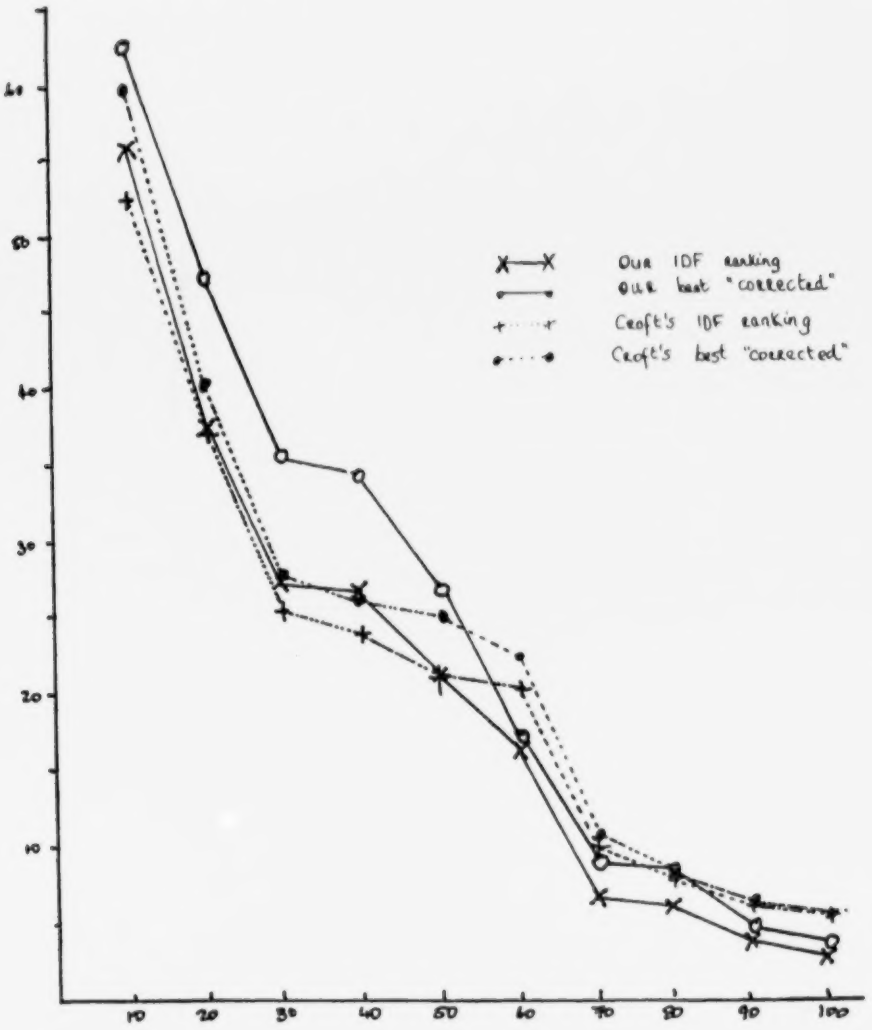

Our IDF ranking
Our best "corrected"
Croft's IDF ranking
Croft's best "corrected"
10
20
30
40
50
60
70
80
90
100

FIG 1.

results are analysed in more detail in the following section.

IDF (Croft)	Corrected (Croft)
52.4	59.6
37.0	40.1
25.3	27.6
23.7	26.3
21.7	24.7
20.0	21.9
9.2	10.2
7.4	7.6
6.0	6.2
5.2	5.2

Table 9: Precision figures for Croft's retrieval strategies

5 Analysis of Results

The precision figures given in the previous section and highlighted in Fig 1 show a quite significant improvement in retrieval effectiveness for all recall values. The fact that our best retrieval strategy which uses syntactic information yields a larger average improvement for the high precision values than Croft's strategy, despite the fact that in our experiment we use syntactic information on the top 50 IDF-ranked documents only, can be attributed to one or two particular queries which are substantially improved at high precision within this top 50 in our strategy. This is due to using a relatively small set of user queries (25) where one or two exceptional results can cause such a shift in the averages.

The significant improvements we have obtained must be attributed to the more judicious use of syntactic information than has been done in work reported elsewhere. It was to be expected that our results would have yielded at least as significant an improvement in retrieval effectiveness as Croft's [85] work. The fact that this expectation has been surpassed indicates that our direction of research is promising and deserves further work. In fact the searching for syntactically derived dependencies from user queries, in the text of filed documents, is very primitive. Making use of sentence and clause boundaries as we have done does not fully utilise the available state of the art techniques in automatic, domain-independent parsing of natural language text (see section 2). Because of this, the experiments reported here must be regarded as only the first step in this direction.

One of the more puzzling points to arise from the results in the previous section concerns the number of relevant documents with no syntactic dependencies in their texts, and the number of non-relevant documents with syntactic dependencies occurring in text. Specifically, from Table 7, what makes the 18 out of 75 documents in the REL/RETR category and 106 out of 171 documents in the REL/NON category, all with no dependency occurrences in their texts, relevant. Also, what makes the 167 out of 415 documents in the NON/RETR category, non-relevant ?

To try and find an answer to the first of these questions, we did a failure analysis on the 18 REL/RETR documents with no syntactic dependence occurring in their texts. Table 10 shows the results we obtained.

The classifications we have used as reasons why REL/RETR documents have no syntactic dependency occurrences, are completely heuristic, but are as accurate as we can use without getting bogged down in detail. All the decisions as to which category a document may belong, are based on our familiarity with the subject area of the documents and queries. The five documents marked as non-relevant and the four documents marked as indirectly relevant are, in our estimation, not directly relevant to the users query as stated. Thus it is reasonable to expect that they have no occurrence of syntactic dependencies in their texts. It makes us wonder, however, how many of the relevant documents in the REL/RETR and REL/NON categories, with some "lucky" syntactic dependency occurrence (possibly noise dependencies) would be non-relevant or indirectly relevant in our estimation.

Non-Relevant	5
Indirectly Relevant	4
Query general, Document specific	3
Relevant	6
Total	18

Table 10: Failure Analysis of REL/RETR Docs with no Dependency Occurrances

The three documents in the "Query general, Document specific" category describe very specific topics while their queries are statements of a more general interest. In these cases, once again, it is not surprising that there are three documents with no occurrence of syntactic dependency, while there are probably many more documents with some syntactic dependency occurrences which would also fit into this category.

A close look at the text of the six documents which are relevant but have no syntactic dependency occurrence shows that this is the case because the document texts make use of synonyms and/or the anaphoric use of pronouns. This situation is inevitable when one considers the way people write text

To summarise the results of Table 10, at least nine of the 75 documents in the REL/RETR category (and we only examined 18) are not relevant to the queries in our estimation, and these are documents with a good deal of search terms occurring in their texts. In addition, we can assume that there are an unknown number of the REL/NON documents, both with and without syntactic dependency occurrence which we would also consider to be non-relevant to their queries.

These results ultimately leads us back to the source of the relevance judgments for the test collection used in the experiment. According to [Fox 83, pp41–43], the relevance judgments are not exhaustive and were obtained in the following way. For each query, the collection was ranked using the Cosine correlation measure, and the top 30 documents were presented to the query originator to obtain relevance judgments. When this was done, 70 more documents were presented to the query originator, these 70 having being obtained by a combination of search methods including using citation and co-citation information on already identified relevant documents, bibliographic coupling, documents being in the same ACM computing review category and documents having the same author as already identified relevant documents, as retrieval criterion. Thus relevance judgments for each query consist of judgments on the 100 documents presented to the query originator, using the above methods.

While the mechanism used to obtain relevance judgments for this test collection are quite methodical, it must be pointed out that they can be misleading. It has been our experience in dealing with the CACM collection that some of the relevance judgments are quite dubious while others appear not to be relevant at all to the users stated information need. If a user asks for information about a topic and judges a particular document

to be relevant to his information need (top 30 in the case of CACM relevance judgments), and that document cites a second document which is presented to the user (case of a document in the next 70 presented for relevance judgment), a user may be inclined to judge that second document to be relevant, meaning relevant to the first document, while what a test collection requires is documents relevant to the original information need, as specified in the query. We believe that cases like this, as well as other possibly misleading relevance judgments, have occurred in the CACM relevance judgments.

One interesting point about these search methods used to obtain relevance judgments is that it is not only possible but probable that documents be presented in this set of 100 documents, which contain none of the original query terms (if retrieved for relevance judgment using the computing review, or co-citation information for example). This would explain why so many of the relevant documents have no occurrence of syntactic dependency, in their texts. As to why 167 of the 415 documents in the NON/RETR category have syntactic dependency occurrence, we can speculate that some of these are "noise" dependencies while the others we are unsure about.

6 Other Work

Before drawing overall conclusions about the results of the experiments reported in this paper, it is worthwhile examining related work in this field. The presentation of this topic has been left until now so that we can interpret our results in terms of other peoples work.

Attempts to include syntactic analysis of text as part of a retrieval mechanism go back over fifteen years to the SMART system. Salton [71, pp207–208] reports attempts to use syntactic analysis methods to recognise syntactically valid phrases in queries and documents for use as indexing units. He also reports methods of recognising phrases from an already existing dictionary of phrases. The conclusions reached in those experiments with respect to incorporating linguistic information in retrieval are that they do not, on average, provide improvements over a standard retrieval using a thesaurus.

At the University of North Carolina the FASIT (Fully Automatic Syntactic Indexing of Text) system is being developed by Dillon, Gray and McDonald. Their early paper [Dillon and Gray, 83] describes an indexing process which assigns syntactic categories to all words in text, extracts content bearing terms/phrases by searching for pre-defined syntactic patterns which indicate content and finally groups synonymous concepts together. Thus the results reported in the above paper indicate no material improvement in retrieval effectiveness using FASIT, when compared to conventional stemming on a very small test collection, but the authors note that the potential for FASIT is far greater than for other methods with respect to refinement and improvement.

A small improvement in the grouping of synonymous concepts aspect of FASIT was reported in [Dillon and McDonald, 83] where FASIT was applied to back-of-the-book indexing. Further work on this topic reported in [Dillon and Gray, 85] led to a reduction in vocabulary size but slightly inferior results compared to earlier FASIT performance. From a linguistic point of view, the FASIT system consists of 75 unique syntactic categories and 1087 disambiguation or grammar rules but despite this the "parsing" of text is quite shallow. For example, sentences are not treated as a whole and a complete parse tree for a sentence, indicating sentence structure and components, is not produced. Thus some grammatically incorrect text would be accepted by FASIT, but when one considers that FASIT, as with all indexing processes, is not supposed to be a sytlistic analysis or grammar checking program, this is quite acceptable.

The overall conclusion on FASIT work reported so far in the literature is that much refinement needs to be done, and FASIT must be applied to a larger test collection of documents before its potential can be reached. The FASIT work has proceeded on the unproven hypothesis that syntactic information can be used to lead to an improvement in retrieval effectiveness, and as the results reported in the earlier part of this paper show, this may be true.

Work done by Bruandet and Kerkouba at the IMAG laboratory in France, and reported in [Bruandet, 85], [Kerkouba, 85] and [Bruandet and Kerkouba, 85] describes how to automatically extract relationships between terms, from free text, using statistical co-occurrance and pseudo-syntactic information. This process is linguistically unsophisticated and computes the strength of relationships between all terms in a vocabulary using a similarity measure based on co-occurrance figures and also using weights assigned to the syntactic category in which each term occurs. For example, the syntactic "weight" assigned to the co-occurrance of two adjectives is eight while the weight assigned to the co-occurrance of two verbs is twenty, and for a verb and a preposition co-occurrance, the weight is five. The values of these weights are completely heuristic. This similarity measure between terms is used to give a term/term matrix where each entry is the strength of association between the terms.

In the paper by [Bruandet, 85], experiments are reported where some maximum complete subgraphs ("cliques") are extracted from this matrix of term/term weights and the paper reports statistics on cliques extracted from a piece of text. The other paper [Kerkouba, 85] proposes an implementation of an indexing strategy which utilises these maximum complete subgraphs derived from a piece of text, to produce single terms, pair, triple, quadruple and quintuple term dependencies as representations for the original text. This indexing process also uses the structure of the document (chapter, section, paragraph and sub-paragraph boundaries) as an integrated component of the indexing process. This suggests that the proposed indexing process would be useful for large, structured documents (books, etc.) only, but the idea of assigning weights to different syntactic classes could be the starting point for an indexing process for smaller unstructured segments of text.

The quality of the indexing process proposed by Kerkouba and Bruandet has been measured against manual indexing and indicates a "good" representation. However the authors note that linguistic tools have not been fully utilised; indeed their implementation of their experiments uses a "semi-automatic lemmatiser" rather than a parser. As is the case with the FASIT system, the IMAG work seems to require further refinement to realise the potential of the early work on the system.

Bartschi, working at ETH in Zurich, has recently described the Caliban system and its retrieval algorithm [Bartschi, 84]. The main emphasis of his PhD was the development of a concept space model for information retrieval. To test out this model he used natural language user queries and processed them into an internal representation using an automatic query extraction method. This method removed stopwords and defined phrases as consisting of adjacent words or words separated by at most two stoplist words except the words AND, OR, OF, FROM, IN and BY. From these phrases, dependencies between words or terms were derived as all combinations of all words in a phrase, without changing word order. Word dependencies were also derived from combinations of phrases, depending on how the phrases were

separated in the original text. All dependencies and single terms extracted from the query using this automatic method were then checked for existence in a thesaurus, and those not present were eliminated.

Retrieval effectiveness result obtained using this indexing method show overall low precision because of the poor linguistic techniques used in analysing the natural language queries. The results also show that if the original natural language queries were short (equivalent to PhraseQ queries?) that including term dependencies as part of the retrieval process led to an improvement in retrieval effectiveness, whereas if the queries were well-formed (equivalent to TextQ queries?) then including term dependencies did not yield improved effectiveness. The central theme of the work reported by Bartschi was to research how the inclusion or exclusion of automatically or manually derived query term dependencies affected retrieval performance, so this work is only indirectly relevant to this paper. Nevertheless, the results obtained with respect to the overall low precision, are worth mentioning.

The paper by Croft [85] discussed earlier, reports experiments where term dependencies (pairs, triples, etc.) have been automatically extracted from Boolean queries, and also where dependencies have been extracted from phrases derived from natural language queries by the user. All these experiments have like ours, been done on the CACM document collection and the dependencies derived from queries were then used in a probabilistic model for retrieval. The dependencies derived automatically from Boolean queries show only a small improvement in retrieval effectiveness.The reasons for this result are that few of the low-scored documents have many query terms and hence would have little modification in their ranking due to the term dependencies, and also because many of the identified dependencies are semantically meaningless.

In the second experiment reported in the Croft paper and compared with our own experiments, term dependencies identified by users themselves by having the users highlight important noun phrases in their queries, have been used. This process could be done automatically and could also be applied to documents as well as queries. The experimental results obtained in the Croft experiment gave significant improvement for longer natural language queries, which fits in with results reported earlier in this paper with respect to TextQ queries. The difference between the Croft work and the work reported here is that Croft uses the user to identify term dependencies, while our work is inclined to having this done automatically, using syntactic analysis, and also our work uses the syntax of the text of documents as part of the retrieval strategy.

The final piece of other work to be mentioned is described in [DeFude, 85]. This paper describes the specification of an information retrieval expert system (IRES). One of the things that this system does is a morphological and syntactic analysis of user queries to identify the important concepts, which the author hypothesises are noun phrases. Verbs in queries are not considered to be content bearing parts of text. Ambiguities are resolved by using the semantics as they are expressed in a thesaurus. In the [DeFude, 85] paper, the system is just proposed and described and no results as regards retrieval effectiveness are provided. Thus the true impact of this work cannot be considered except to point out that the authors hypothesis that only noun phrases are content bearing, seem to be unfounded but readily accepted in the literature.

To sum up the research work done elsewhere which uses syntactic information as part of an overall retrieval strategy, no significant improvements have ever been obtained with the exception of the results in Croft [85] and in this paper.

7 Conclusions

In this paper we report an analysis of the use of parsing text as part of a retrieval strategy for text documents. For this we use a test collection of documents and queries. We performed a statistically-based retrieval on this collection of documents and followed this with a syntactic analysis of the text of the retrieved documents and the relevant documents not retrieved, searching for dependent terms automatically identified from the query. This analysis has given us some information on the syntax and distribution of these dependencies among document texts. Based on this analysis we then proposed and implemented a retrieval strategy which incorporates some syntactic processing as part of retrieval. This strategy gave us a significant improvement in retrieval effectiveness when compared with our benchmark results which do not include syntactic information.

Over the last twenty years the experimental results obtained when incorporating syntactic information into an overall retrieval strategy have seen few significant improvements. In previous work researchers have tried searching for syntactic patterns [Dillon, 1983], using specific syntactic categories to determine dependencies [Bruandet, 1985 and Kerkouba, 1985], determining phrases in the context of a list of known stopwords and adjacency constraints [Bartschi, 1984], and other similar approaches, but it is clear at this point, that these approaches are wrong. What these unsuccessful approaches do have in common is that they use syntactic information for indexing documents and not in the context of any specific query. The only work to have yielded significant improvement by including syntactic information [Croft 85 and this paper] does this at retrieval time and in terms of a particular query. It now seems that this is a worthwhile direction in which to proceed.

Our future work on incorporating syntactic information into retrieval will continue by investigating paying more attention to word order, performing more detailed processing on the text of filed documents and incorporating information on specific syntactic categories of words in documents and queries.

Acknowledgements

The work reported in this paper was carried out as part of ESPRIT 4.4.2.1 Project No 59 (Project Minstrel). The author would like to acknowledge the help given by other project members Bruce Croft, David Harper and Keith van Rijsbergen. In particular, Bruce Croft and David Harper provided the initial idea for the experiments and Bruce Croft supplied some of his own experimental results and many fruitful discussions.

References

[Bartschi, 84] *"Term Dependence in Information retrieval Models"*, M.A. Bartschi, PhD. thesis, ETH Zurich, 1984.

[Bruandet, 85] *"Modele Partiel de Connaissances Pour un Systeme de Recherche D'Informations"*, M-F Bruandet, Proceedings of RIAO85, Grenoble, 1985, pp101-114. (in French, English version available from author)

[Church and Patil, 82] *"Coping with Syntactic Ambiguity or How to Put the Block in the Box on the Table"*, K Church and R Patil, MIT Technical Report, MIT/LCS/TM-216, April 1982.

[Croft, 85] *"Boolean Queries and Term Dependencies in Probabilistic Retrieval Models"*, W.B. Croft, Journal of the ASIS (in press), 1985.

[DeFude, 85] *"Different Levels of Expertise for an Expert System in Information Retrieval"*, B. DeFude, Proceedings 8th ACM SIGIR Conference, Montreal, June 1985, pp147-153.

[Dillon and Gray, 83] *"FASIT: A Fully Automatic Syntactically Based Indexing System"*, M. Dillon and A.S. Gray, Journal of the ASIS, 34, 1983, pp99-108.

[Dillon and Gray, 85] *"Improved Results using FASIT for Associative Retrieval"*, M. Dillon and A.S. Gray, (private communication, Jan 85).

[Dillon and McDonald, 83] *"Fully Automatic Book Indexing"*, M. Dillon and L.K. McDonald, Journal of Documentation, 39, 1983, pp135-154.

[Fox 83] *"Characteristics of Two New Experimental Collections in Computer and Information Science Containing Textual and Bibliographic Concepts"*, E.A. Fox, Cornell University Technical Report 83-561, 1983.

[Grishman, 84] *"Natural Language Processing"*, R. Grishman, Journal of the ASIS, 35, 1984, pp291-296.

[Hayes and Carbonell, 83] *"A Tutorial on Techniques and Applications for Natural Language Processing"*, P.J. Hayes and J G Carbonell, Carnegie-Mellon University Technical Report, CMU-CS-83-158, 1983.

[Kerkouba, 85] *"Indexation Automatique et Aspects Structurels du Texte"*, D. Kerkouba, Proceedings of RIAO85, Grenoble, 1985, pp227-249. (in French, English version available from author).

[Kerkouba and Bruandet, 85] *"Automatic Indexing Method Using a Partial Knowledge Model for an Information Retrieval System"*, D Kerkouba and M-F Bruandet, private communication, 1985.

[Kraft, 85] *"Advances in Information Retrieval: Where is that #*"*&@$ Record"*, D.H. Kraft, in:Advances in Computers, Vol 24, M.C. Yovits Ed., Academic Press, 1985.

[Marsh and Friedman, 85] *"Transporting the Linguistic String Project System from a Medical to a Navy Domain"*, E Marsh and C Friedman, ACM Transactions on Office Information Systems, 3, 1985, pp121-140.

[Porter, 80] *"An Algorithm for Suffix Stripping"*, M.F. Porter, in "New Models in Probabilistic Information Retrieval", British Library RandD Reprt no. 5587, University of Cambridge, 1980.

[Salton, 71] *"The SMART Retrieval System: Experiments in Automatic Document Processing"*, G. Salton Ed., Prentice-Hall: New Jersey, 1971.

[Salton and McGill, 83] *"Introduction to Modern Information Retrieval"*, G. Salton and M. McGill, McGraw-Hill, 1983.

[Salton, 85] *"A Note on Information Retrieval Models"*, G. Salton, Proceedings of RIAO85, Grenoble, 1985, pp1-27.

[van Rijsbergen, 79] *"Information Retrieval"*, 2nd Edition, C.J. van Rijsbergen, London: Butterworths, 1979

[Wilensky and Arens, 80] *"PHRAN - A Knowledge-Based Approach to Natural Language Analysis"*, R. Wilensky and Y Arens, University of California Memo, UCB/ERL/M80/34, 1980

[Winograd, 83] *"Language as a Cognitive Process, Volume I Syntax"*, T. Winograd, Addison Wesley, 1983.

Appendix: Heuristics for Deriving Dependencies from Queries

The model of natural language used in the heuristics for deriving dependencies from queries is a very simple and straightforward one. Thus the resulting heuristics are crude in places and could certainly be improved upon, with more experience and further work in the area. Basically, we presume that queries consist of a series of independent noun phrases (NPs), a series of conjoined NPs independent NPs or combinations thereof, or grammatically well-formed textual statements. To derive dependencies from natural language queries we perform four steps:

- Step 1: Turn grammatically well-formed text into a sequence of conjoined and independent NPs.
- Step 2: Derive dependencies from within each NP.
- Step 3: Derive dependencies from the conjunction aspects of NPs.
- Step 4: Remove any duplicate dependencies.

The heuristics used in the first three of these steps will be described in turn.

Step 1, turning text into NPs can be done by one of many available parsers. In the experiments reported in this paper, parsing of user queries was done by hand.

In our simplistic view of the English language, a noun phrase consists of

- One or more headnouns.
- One or more adjectives.
- One or more following or preceding modifying nouns,
- which themselves may be further modified recursively.
- One or more following qualifying NPs or Prepositional Phrases (PPs)

To derive dependencies from within a single NP we use all possible combinations of adjectives and headnouns from within that NP. For example, if a NP is of the form

adj1 adj2 headnoun

then the dependencies from that NP are

(adj1 adj2) (adj1 headnoun) (adj2 headnoun) (adj1 adj2 headnoun)

This rule is a good example of our philosophy of deriving as many dependencies as possible within reason, from queries and accepting that some of them may be semantic nonsense. Using the above rule the NP "big green apple" would give the dependencies

(big green) (big apple) (green apple) (big green apple)

all of which appear to be semantically correct, but the NP "pale green apple" would yield the dependencies

(pale green) (pale apple) (green apple) (pale green apple)

the second of which is semantic nonsense. Our other heuristic rules are full of further examples of this kind of situation. Our reason for tolerating this is that if the dependencies are semantic nonsense and not content bearing, then they should not occur in document text. The semantically nonsensical dependencies are what we have termed "noise" dependencies earlier in the paper.

If a NP has more than one headnoun and also some adjectives, and it is syntactically ambiguous as to which of the headnouns the adjectives refer, we derive dependencies as if the adjectives refer to both headnouns, though this may not always be the case. For example, in query number 23 we have

"distributed computing structures and algorithms"
adj1 adj2 headnoun1 headnoun2

In this case it is syntactically ambiguous whether the query refers to distributed computing structures and also algorithms, or distributed computing structures and distributed computing algorithms, or even distributed computing structures and distributed algorithms ! The dependencies we would derive from this NP using our heuristics are

(distributed computing structures) (distributed computing algorithms) (distributed structures) (distributed algorithms) (computing structures) (computing algorithms) (distributed computing) (structures algorithms) (distributed computing structures algorithms)

For deriving dependencies from the conjunction aspect of say two NPs, we form dependencies consisting of the union of all terms in the two NPs and a set consisting of the headnouns from the NPs. Also, if any of the NPs have adjectives then all headnouns plus all permutations of adjectives give more sets of dependencies. For example, a query with the surface structure

adj1 headnoun1 adj2 headnoun2

would give

(headnoun1 headnoun2) (adj1 headnoun1 headnoun2)

(headnoun adj2 headnoun2) (adj1 headnoun1 adj2 headnoun2)

but not (adj1 headnoun2) or (headnoun1 adj2).

For the case of qualifying prepositional phrases (PPs), what we do in this case is remove the preposition and we are left with a noun phrase, which we treat as conjoined with the NP it originally qualified. For the case of modifying NPs which themselves in turn may be modified by other NPs, we treat the modifying NP as a NP conjoined with the NP it originally modified, but the modifying NP is not conjoined with any secondary NPs the originally modified NP may have been conjoined with. For example

NP1 (modNP1) NP2 would give (NP1 NP2)

(Np1 modNP1)

but NOT (modNP1 NP2), while

NP (modNP1(modNP2)) would give (NP modNP1)
(modNP1 modNP2)

but NOT (NP modNP2).

CALIN - A USER INTERFACE BASED ON A SIMPLE NATURAL LANGUAGE

by

P. BOSC, M. COURANT, S. ROBIN

IRISA/INRIA et UNIVERSITE de RENNES
CAMPUS de BEAULIEU
35042 RENNES CEDEX
FRANCE

ABSTRACT

In the framework of an application dealing with classified advertisement matching, a dedicated user interface has been designed and implemented. Its major originality relies on the user's language which is neither an artificial one, nor the usual natural language, but in fact the ad language. Beyond the language itself, the interface provides some facilities such as paraphrasing or explanations when needed. An expert system approach has been adopted and the interface is built up from the knowledge given by experts. They are in charge of describing what are acceptable ads, from both syntax and semantics points of view ... Although designed in the context of ad matching, that interface may interestingly be adapted to other retrieving systems. We especially think that an ad-like language is well-suited to ask questions since it is based on natural simple expressions. A given sentence involves terms that stand for elementary conditions applying to instances of a logical object contained inside the information system. This approach defines a complete interface, involving both a language and aiding capabilities. Moreover, the query language, although less powerfull, represents a compromise between artificial languages and the usual natural language, with respect to ergonomics and analysis complexity.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

1 - INTRODUCTION

The design of "natural" language interfaces for the final users is one of the claimed objectives for the fifth generation computers. User interfaces aimed at data retrieval may be classified into two groups. The first one provides users with languages strongly connected to the representation universe, whereas the other allows the user to talk about data as they appear in the reality [BUBE 82]. Most often, artificial languages are associated with the first kind, such as SEQUEL [CHAM 76] or G-WHIZ [HEIL 85] for database querying. The typical basis for the second kind of interfaces is "the natural language", [HEND 76] or [CODD 78] for example.

If artificial languages are easy to analyse, their use is not obvious for casual users. On the other side, the natural language is easy to use but reveals to be highly complex to parse, even in the restricted case of queries. Consequently, only a limited number of syntactical structures are allowed. Moreover, queries are not concise and often ambiguous what leads to numerous additional dialogues.

At last, we must say that the language is not the only component of the interface. Especially, it is mandatory to inform the user about the meaning of the query, from a system point of view. Then, if the user does not agree on this interpretation or if the system rejects the query, the interface must help him to express a new right query. This latter phase is founded on justifications and explanations provided by the interface through dialogues.

Facing the lack of a universal well-agreed interface, we have been led to design an *ad'hoc* interface for a given application, especially aimed at an easy and natural way of asking questions.

This work is partially supported by the French Project dedicated to the third generation databases (PRC BD3).

In our case, the first objective was a system to manage classified advertisements (ads), called HAVANE [BOSC 83], [BOSC 85]. The major point of this system is to allow users to question an ad database *by means of a query, which is itself an ad.* Moreover, we desired to provide users with pretty agreable capabilities, such as a well-suited paraphrase of his ad and explanations to solve dissensions. Consequently, a dedicated interface, based on artificial intelligence mechanisms, was designed and implemented. In fact, the main interest of this part of our work is an exhaustive investigation of a specific application in order to isolate exactly which knowledge is required.

Moreover, we got gradually convinced that the approach was adequate to build other retrieval interfaces. The principle is to generalize the type of query language and the related dialogues. Although it does not support any kind of query, such an interface, called CALIN for classified advertisement like interface, is likely realistic for many applications.

This paper focuses on the specific interface that has been designed and implemented. In section 2, the principal roles of the HAVANE interface are described and so, we see that such an interface may be thought of as an expert system. Section 3 is devoted to a characterisation of the ad language. From its specificity it emerges that a particular way of analyse may be employed, what makes the originality compared with the usual analysing techniques. Then, in section 4 we detail the knowledge that the expert is asked about to specify a given domain of ads. The specification of the user's language constitutes the main part and involves a syntactical and a semantic component, both based on attribute grammars. Section 5 deals with the implementation features of the interface. The rules provided by the experts *do not define all the parts* of the interface. Especially, the analysing strategy is not part of the experts rules and we focus on this topic usually called the *inference engine*. The next section gathers several examples aimed at an overview of the exchanges between the system and the user. The extension of this approach to other retrieving systems is discussed in section 7. We particularly point out the notion of *object* which is the user's view of data contained inside the system.

2 - DESCRIPTION OF THE HAVANE INTERFACE

2.1 - Overview

HAVANE is basically intended for the interrogation of an ad database (real estate, cars, jobs, ...), by means of an ad. In fact, given an ad, HAVANE will perform a first selection through the base, but the user is in charge of making his final choice. This approach supplies several advantages, especially since no learning and no information about the data structure is necessary. Beyond this first goal, the interface provides the user with some help whenever problems arise. At last, it is to mention that the user is able to store his ad inside the database. In such a case, the ad is not only a question but it becomes an object of the database. Any stored ad consists of two parts : the user's text and the related internal formatted representation issued from the analysis. The users do not see this representation, which is used by the matching process. They only deal with the texts of ads, either when they enter a query or when they receive its answer.

We now examine the various functions devolved to that interface. We finally see that the nature of these functions led us to build HAVANE as a knowledge-based interface.

2.2 - Classified advertisements analysis

The properties of the ad language will be detailed in section 3. Nevertheless, we can already point out some analysis requirements from a user point of view.

First of all, the user must be allowed to mention elements which are of little importance from a semantic viewpoint. For instance, the expression: *chimney in living room* or *wooden fence* may be considered as superfluous by the system designer. Such expressions must be ignored during the analysis, by if the ad is stored and then returned to another user, he will make some use of these elements to perform his final selection.

The second role of the analysis is to detect *inconsistent* ads. An ad is said inconsistent if either it contains a contradiction between some of its elements -for example a house for hire and an amount which is a selling price-, or it violates legality rules -racial discrimination-, or mandatory information is missing -the location of a house or appartment to sell, for instance-.

As often with natural language sentences, an ad may be ambiguous, even if such cases are unusual. Consequently, the analysis *must produce all the comprehensions* tied to a given ad.

If some pieces of information are mandatory, others may be omitted by the user and assumed by the interface. For example, the ad : *proposes cottage 5 rooms, garden and garage, at Nice* may be interpreted as a selling proposal rather than a hiring one, according to a conventional practise. This kind of uncompleteness looks like *default options*.

The last point handled by the analysis deals with spelling mistakes tolerance. Since the system is intended for non specialists, this point is of prime importance. Thus, the word *cottage* must be found even if the user wrote it *cotage* -actual spelling mistake- or *cittage* -typing mistake since o and i are neighbour on the kayboard.

2.3 - Paraphrase generation

When the analysis succeeds, its results are given to the user by means of paraphrases. They constitute a way to tell the user about what has been understood and to beg his agreement. The opinion of the user is the more necessary as ads are ambiguous.

The paraphrase can be something else than a simple rewriting of the initial text. For example, if some element is not explicitly stated in the ad, but has been *deduced in some way*, it could interestingly appear in the paraphrase.
Consider the text : *house 5-6 rooms near Paris 1000000 francs*. We can guess that the system will be able to understand that the user wants to buy a house, due to the presence of *1000000 francs*. Consequently, the paraphrase could be : *you desire to buy a house ...*

Nevertheless, it can be of interest for complicity purpose, that some of the locutions expressed by the user are incorporated into the paraphrase. For instance, the ad : *to sell cottage ...* could be better paraphrased : *you have a cottage for sale ...* than *you have a house for sale ...* even if *cottage* and *house* are assumed equivalent for the internal level.

2.4 - Dialogues and explanations

The third function of our interface concerns the dialogues. They will take place when either the interface rejects the user's ad, or the user does not agree on the paraphrase. In the first case, the ad must not only be refused but some explanations about the rejection -which information is missing or which expressions are conflicting, ...- must be given. Moreover, the user must be allowed to rectify so that the analysis may be resumed. The second situation is a little bit more complex since the interface does not know the reason of the dissension. We distinguish several cases, among them :

- the user realizes that he forgot to express something, for instance the size of a house,
- some element given in the text is not reflected in the paraphrase, for example the presence of a chimney since it has been considered as superfluous by the system designer,
- there is a contradiction between the paraphrase and the user's meaning, due to a *wrong deduction* or *assumption* performed by the system.

A first step will determine the cause of the dissension and then the right interpretation will be searched.

2.5 - Database retrieval

When the user agrees on the paraphrase, the retrieving process takes place. Since an ad is not directly a query, the interface has to derive the actual query which will be addressed against the database. This query will refer to the internal representation of ads and its result must be a reasonable number of ad texts. In that sense, the user gets a high-level answer with respect to a list of attributes for example.

To achieve our goal -i.e. a limited number of ads-, the retrieval is based on the proximity of values and fuzzy querying mechanisms [CHAU 85]. So doing, the result is interpreted as the closest ads of the database tied to the user's ad. This point is not detailed in this paper since the specification of this processing is very particular for that application. Nevertheless, it is part of the interface in accordance with the approach suggested in [BROV 85]. The basic mechanisms related to ad matching -or comparison- are developed in [CHAU 83].

2.6 - Conclusion : our approach

As we can seen, this interface requires knowledge about the terms or expressions used in the ad language, what is a consistent ad, what can be *deduced*, how to define the advertisements to be delivered as an answer, and so on. Moreover, the dialogues lie on the justification of the reasoning. At last, the analyser must be able to acquire information during the dialogues to resume its reasoning. These are some of the reasons which led us to design HAVANE as an expert system [LAUR 82]. In fact, several experts are likely necessary to specify all the knowledge. The definition of the language of a given domain -cars, ...- and the principles of the retrieval are dependent on distinct competences.

3 - THE CLASSIFIED ADVERTISEMENT LANGUAGE

We now present the main features of the ad language. Although it would have been possible to treat it as a pure subset of the natural language, its specificities guided us to an original analysis strategy.

3.1 - Characteristics

From an ergonomic viewpoint, the ad language presents the advantage of being a *familiar easy to use* language. With respect to natural language, a sentence is not based on complex syntactical and grammatical rules and structures. So, an ad is more convenient to write than a usual sentence. Especially, the elision of articles, pronouns, ... causes a great concision. The interest of such a feature including shorthands and abbreviations has already been pointed out earlier [WOOD 77]. At last, since ambiguities are rare although possible, dialogues are expected to be limited.

We now examine the language from a linguistic viewpoint. First of all, we must observe the lack of an overall syntactical structure. Most often, an ad is built as a set of syntactical units that are independent, optional and order-free. Moreover, connection words like articles, conjunctions and prepositions are often omitted.

The second point concerns the presence of what we call *strong words or expressions*. Their interest resides in the fact that a concise locution informs about several items. For instance, the expression : *2000 f max* suggests that it is a request -since *max* is used-, for hiring because of the amount *2000 f*. These strong words will lead to *numerous inferences*.

Lastly, we found out that most of the time the ads are not ambiguous. Such cases occur when an expression may apply to distinct items -10000 km for instance is either related to the engine or the brakes of a car- or if a string accepts

may be introduced. Moreover, a description does not involve any operating statement. Operating choices are devolved to the inference engine -cf 5-.

4.1.2 - The syntactical analysis grammars

The SAGs define elementary syntactical structures whose semantics is captured by means of attributes. The SAGs shown below relate to the previous example.

«nbrooms(N,N)» --> «type» «number(N)» | ...

«type» --> type | F | T

«nbrooms(N1,N2)» --> «number(N1)»
«number(N2)» | ...

«habitationname("h")» --> house

except near house | in house | ... (7)

«legcase$_1$("yes")» --> black not allowed | ... (8)

Inside SAGs, the order of right side elements is meaningful and implies the order of the derived substrings in the ads. A SAG fits with elementary information appearing as contiguous strings. When separate substrings make up an information, several SAGs are requested. At last, let us recall that SAGs are intended for the recognition of useful strings with respect to the application, including forbidden locutions -8- or words in an awkward context -7-.

The experts are asked no special knowledge to deal with spelling mistakes. They are treated on the basis of general transformation rules and this topic is no longer discussed.

4.2 - The paraphrasing knowledge

The paraphrase building mechanism is twofold. Paraphrasing elements are tied to both SAGs and inference rules and a composition expression is associated to the root of the SD. Consequently, attributes are added to bear the paraphrase.

For instance, we shall have :

«nbrooms(N,N"involving" N "rooms")» -->
«type» «number(N)» | ...

«habitationname("house","a house")» --> house

«habitationname("house","a cottage")» --> cottage

In the context of the inference denoted 3 above, the paraphrasing attributes PTM and PTS will appear in the left side membre - ad(...) - and we shall have the inference :

AMNT1 < 3000 *and* ... --> TS = "request" *and*
TM = "hiring" *and*
PTS = "you desire" *and*
PTM = "for hire"

Presently, the paraphrasing elements are only combined through the concatenation denoted +. So, the final paraphase formula in the example would be :

«ad(...), PARAPH)» --> ...
with PARAPH := "we understood that" + PTS + PNAME
+ PTM + ...

4.3 - The explanations

In fact, the explanations are essentially based on the knowledge supplied for the analysis and the results of the analysis. The unique additional information concerns a *meta vocabulary* to define an external name for each SAG, each attribute, each attribute value, ... These names are intended to build agreable meaningful messages from a user viewpoint. For instance, the nature of the habitation -called NAT- must have a value according to rule 5 of the example. If the expert gives NAT the external name : *the nature of the habitation* , the lack of this information in an ad will causes the message : "the nature of the habitation has been omitted, you must choose among *house, apartment, ...*". *house, apartment* are external representations for the internal values taken by the attribute NAT.

5 - THE IMPLEMENTATION

We now give the guideliness of the knowledge use. The analysis strategy is first briefly described. This topic is more detailed in [COUR 85]. We then overview the situations ending up to dialogues and the bases of the returned explanations.

5.1 - The parser

The parser is aimed at producing all the comprehensions issued from an ad text. Moreover, they will be classified according to their relevance.

To do so, all the SAGs are activated against the text and their results are applied to the Structure Descriptor to obtain compatible subsets of SAGs. The relevance of the comprehensions is measured according to the use of textual components of the ad. Our strategy privilegiates the richest compatible combinations of analysed substrings. The inference mechanism influences also the arrangement in so far as additional information is generated through an inference.

The basic reasons of the multiple comprehensions are on the one hand syntactical conflicts -the same substring or two overlapping substrings are recognized by at least two SAGs or different substrings are analysed by a same SAG- and on the other hand the possible *non determinism* inside the rules appearing in the Structure Descriptor.

As the parser works, it builds the paraphrase tied to the current comprehension. When all comprehensions are produced, the paraphrase connected to the most relevant comprehension is first sent to the user. We shall see in the next subsection that the other comprehensions will be useful if the user disagrees on that first paraphrase.

5.2 - The explanation process

Let us recall that dialogues take place when

several decompositions -*garden 200sqm habitable* has two meanings : the surface of the garden is 200 sqm and the house is habitable or there is a garden and the surface of the house is 200sqm.

3.2 - Analysis principles

Beyond the *objective points* reported above, an additional consideration had an effect upon the analysis design . It is of no use to parse elements of ads related to *subjective descriptions*. Expressions such as *located in a nice wooded place* cannot be taken into account for ad matching, since not quantifiable.

Consequently, instead of an exhaustive analysis of the ads, the strategy is founded on the recognition of *useful strings*. This method is rather original, although suggested in [LUBO 85] in an other context. If the elementary syntactical components are simple, the overall semantics of an ad cannot be captured by a pure *key-word oriented* analysis. First, it is important to capture the terms in their context. So, the key-word *house* usually stands for the object of the transaction, except if it appears in *near the house of parliament*. Such an occurrence of *house* must be picked up in order to avoid what we could call *a misunderstanding*. Another point concerns the application requirements. For instance, one can be interested in the deduction of a global size of a house starting from a detailed description like : *house involving kitchen, living, 2 rooms on 1st floor, 3 rooms on 2nd floor*. A last argument against a pure key-word analysis, probably the most important, is the fact that we want to deduce new information and to check some coherence inside the user's text. For instance, the expression : *to sell house ... monthly 2000 f* must lead to a contradiction, since *monthly 2000 f* expresses a hiring transaction, whereas the explicit verb indicates a selling one.

The advantages of a partial analysis are the simplicity of the analyser and the need of a limited knowledge. The counterparts are on the one hand, the occurrence of dissensions since elements are not recognized and thus paraphrased and on the other, the possibility for jokers to write ads which actual meaning is far from what the interface understands. However, for the latter case, we bet on a *good use* of the system and for the former, the dialogues will advise unexperienced users.

4 - THE REQUIRED KNOWLEDGE AND ITS REPRESENTATION

In the previous two sections, we saw the expected behaviour of the system and the properties of the user's language. We now inspect what experts must give and how their knowledge is represented through formalisms [LAUR 82].

4.1 - The analysis

The required knowledge for the ad understanding is structured according to two levels. The first one concerns the syntax of the basic components, the other deals with the overall semantics of an ad -inferences, consistency, ...-. These two levels are expressed by means of a same formalism : *attributed context-free grammars* [KNUT 71]. The *structure descriptor* -SD- is used for the semantic level whereas *syntactical analysis grammars* -SAGs- specify the basic allowed expressions.

4.1.1 - The structure descriptor

The role of the structure descriptor is to describe an ad as a tree. Inference and consistency rules are tied to the nodes. The leaves are the axioms of the SAGs what establishes the connection with the syntactical level. The attributes tied to the root build up the internal representation of the ad. The formalism involves most of the possibilities provided by object-oriented languages. It is especially possible to compute attribute values, to constrain these values and to affect default values. Moreover, attribute grammars are adequate to check the coherence of the specifications [DERA 83]. The axiom of the SD is named ad, the non terminals are denoted ⟨..⟩ , the terminals, denoted ⟨⟨..⟩⟩, are the start symbols of the SAGs. Lastly, it is to notice that the order of the right side non terminals is not significant and that left side attributes are synthetized from right side attributes.

Let's consider the following example concerning the real estate domain :

⟨ad(TS,TM,N1,N2,...)⟩ --> ⟨ transaction(TS1,TM1)⟩
⟨ object(NB1,NB2,NAT,...)⟩
⟨ exchange(AMNT1,AMNT2,
AMNT3,...)⟩
⟨ legality ⟩

with
(1) TS1 = *nil and* TM1 = "buy-or-sale"
--> TS = "proposal"

(2) TS1 = *nil and* (TM1 = "hiring" *or* NB1 ≠ NB2)
--> TS = "request"

(3) AMNT < 30000 *and* (AMNT1 ≠ AMNT2 *or*
AMNT3 = "max" *or* "min" *or* ...)
--> TS = "request" *and* TM = "hiring"

(4) *not*(NB1 ≠ NB2 *and* TS1 = "proposal")

(5) *not* (NAT = *nil*)

⟨legality⟩ --> ⟨⟨legcase$_1$(l_1)⟩⟩ ...
⟨⟨legcase$_n$(l_n)⟩⟩
with
(6) *not* (l_1 = "yes" *or* ... *or* l_n = "yes")

Expressions 1, 2 and 3 express default options. More precisely, they are intended to tell possible values for the transaction type -request or proposal-, when it is not explicitly stated in the ad, depending on values taken by other attributes. Consistency constraints are shown in expressions 4, 5 and 6 with a contradiction -for a proposal the room number must be precise-, a mandatory information -the nature of habitation NAT must be stated in the ad-, and a legality checking -none of the forbidden expressions recognized by the SAGs legcase$_i$ must appear-.

This description mechanism is close to production rules [LAUR 82], and, then non determinism

the system does not succeed in understanding the ad or if the user disagrees on the proposed paraphrase.

In the first case, if a contradiction is detected, we go further back to the responsible textual elements. The user is then asked to modify one of the conflicting elements. If a mandatory information is missing, the lack is signalled to the user. The message demands a choice among the possible values of the attribute.

The second case is more complex to handle. In fact, the dissension is due to either a missing or an extra element inside the paraphrase. The investigation of this point revealed that the various reasons could be easily distinguised one from another. Here again, we are sometimes led to go back to *wrong inferences* -at the user's sense-. The real key of the problem resides in the availability of all the comprehensions. This allows to know whether a missing information appears in an other comprehension or not.

6 - SOME INTERACTION EXAMPLES

The role of this section is to give an idea about the exchanges taking place between the system an a user. Several cases are illustrated in the following examples. All of them refer to the real estate universe. The examples are supposed to be self-explanatory and we do not describe the related knowledge. At last, let us notice that the ads are given according to their french formulation.

EXAMPLE 1

```
PLEASE, TYPE YOUR AD ...

to sell 3000 f, nice villa at Cannes near seashore.

A CONTRADICTION HAS BEEN DETECTED SINCE

(1) FROM "SELL", WE DEDUCED A "BUY-OR-SALE"
    TRANSACTION
(2) FROM "3000 f", WE DEDUCED A "HIRING"
    TRANSACTION

WHICH ITEM DO YOU CHANGE (1/2) ?

2

PLEASE, GIVE ITS NEW VALUE ...

3000000 f
```

Then, the analysis may be resumed and a paraphrase of a new interpretation will be given to the user.

EXAMPLE 2

```
PLEASE, TYPE YOUR AD ...

to sell cottage 6 rooms, near rennes, garage.

WE UNDERSTOOD THAT YOU DESIRE TO SELL A COTTAGE
INVOLVING 6 ROOMS LOCATED NEAR RENNES.

DO YOU AGREE ?

no, why not garage ?

WE DON'T RECOGNIZE THE WORD "GARAGE".

IS IT CONNECTED TO ONE OF THE FOLLOWING ITEMS ?

   (1) NATURE OF THE HABITATION
   (2) TRANSACTION MODE
   (3) TRANSACTION TYPE
   .
   .
   .
   (9) PRICE ?

no

SORRY, DO YOU ACCEPT OUR UNDERSTANDING ?

yes

WE ARE NOW LOOKING FOR THE TEN BEST ADS
```

EXAMPLE 3

```
PLEASE, TYPE YOUR AD ...

house at nice for life.

WE UNDERSTOOD THAT YOU DESIRE TO SELL A HOUSE
LOCATED AT NICE FOR LIFE

DO YOU AGREE ?

no, why desire to sell ?

FROM "FOR LIFE", WE DEDUCED A "BUY-OR-SALE" TRAN-
SACTION.
THEN, SINCE THE TRANSACTION TYPE WAS MISSING AND
THE TRANSACTION MODE WAS "BUY-OR-SALE", WE DEDUCED
A PROPOSAL.

DO YOU WANT TO MODIFY YOUR AD ?

yes

WHICH ITEM DO YOU CHANGE ?

   (1) NATURE OF THE HABITATION
   (2) TRANSACTION MODE
   (3) TRANSACTION TYPE
   .
   .
   .
   (9) PRICE ?

3

PLEASE, GIVE ITS NEW VALUE

   (1) REQUEST
   (2) PROPOSAL

1
```

EXAMPLE 4

```
PLEASE, TYPE YOUR AD ...

urgent house 3000 f monthly garden 250 sqm habi-
table near school
```

WE UNDERSTOOD THAT YOU PROPOSE FOR HIRE A HOUSE FOR 3000 F PER MONTH WITH A GARDEN AND 250 M2 HABITABLE SURFACE

DO YOU AGREE ?

no, why habitation surface 250 m2

WE HAD TWO POSSIBLE INTERPRETATIONS FOR 250 SQM

(1) GARDEN AND HABITABLE SURFACE 250 M2
(2) GARDEN 250 M2 AND HOUSE HABITABLE

DO YOU AGREE ON THE INTERPRETATION NUMBER 2 ?

yes

WE ARE NOW LOOKING FOR THE TEN BEST ADS

7 - EXTENDING THE APPROACH

As stated in the introduction, the initial context of our work is the ad matching application. Consequently, our method surely works for any ad domain. However, it is obviously of interest to examine if the approach may apply to a wider scope of retrieval problems. The idea is not to replace the existing interfaces, but rather to propose an alternate way of interrogation when it is possible and more convenient for users.

7.1 - The language

From a user point of view, the interests of that kind of language are :

- its independence with respect to the representation and structure of data inside the information system,
- its structural simplicity since very few syntactical structures are used and the order of the different terms does not matter,
- its concision since few expressions are used to tell all the desired information in a given sentence,
- its suppleness because extra terms are allowed.

Presently, it is able to express a conjonction of conditions applying to a given implicit set. It involves no special feature in order to specify the desired result -in our application, the text tied to the selected ads. So, our interface may be directly used for applications that do not need additional capabilities.

A first class of applications concerns leisure or travel information systems. For example, it appears natural for someone to state *train for tomorrow, Rennes Roma, with sleeping, arrival before 10 a.m.*, if he looks for a night train starting tomorrow from Rennes and which arrival to Roma is before 10 a.m. The interpretation is straightforward and the expected result is two or three detailed routes, including timetables to reach Roma from Rennes.

Such an approach may also be considered in the scope of automated offices. For instance, it would be easy to fix appointments between persons by means of messages like : *rendez-vous with Jones monday 1st 2 hours max.*

7.2 - The notion of object

In all these cases, the contents of the system are perceived through a very simple way by the user -set of ads or routes, ...-, what we call *objects*. A given sentence of the language stands for a selection among the instances of the considered object. But, very often, the semantics of the data contained in the system is complex, and even if a user deals with a given subset, it is not possible to modelise this subset as *one object*. For instance, in the context of a database containing suppliers and supplies, we do not think it is natural to gather them inside one object. Such a modelisation is achieved by *the universal relation approach* [KORT 84], [ULLM 83].

So, we suggest that several objects can be seen by a user. Each object is a semantic entity, that makes sense for the users. It is defined by the expert and establishes connections with the actual stored entities -or relationships. Consequently, the language must be extended in order to specify *which object is concerned by the query and the desired results* when they are not implicitly defined by the application. A possible version of the user's queries could be :

⟨desired results⟩ for ⟨object⟩ with ⟨qualification expression⟩

where ⟨desired results⟩ is the list of items names, ⟨object⟩ is the name of the concerned object and ⟨qualification expression⟩ is an ad-like expression. In fact, the actual form of the query will be choosen by the expert who will be in charge of designing each specific interface. So, we must slightly modify the specification formalism to introduce the two new components of a query.

The last extension concerns the generation of the internal query addressed to the information system. Right now, the result of the analysis consists of a set of attribute values. This solution suffices for our application, since the HAVANE interface is not connected to any information system. We think that there is no real problem to replace these attributes by a query expressed in the adequate query language. This query will be constructed in the same way as the paraphrase.

This approach does not offer all the querying capabilities provided by usual languages, but it is expected for a wide class of use. For instance, it is not planned to allow a question stated verbally as : *find the employees earning more than at least three managers, each of which has more than 20 employees in this department.*

8 - CONCLUSION

This paper presents an interface for classified advertisement management. Then, some kind of generalization is discussed.

The interface is based on the use of the ad language found in the newpapers. First of all, the functions offered to users are pointed out. They mainly aim at the understanding of an ad, the elaboration of a paraphrase which is a basis for agreement and discussion, the supply of explanations, the retrieval of the most relevant ads out of the base.

Then, the properties of the ad language are examined. From a user point of view, the interest resides in its simplicity. From a linguistic point of view, its concision and its weak syntactical structure are undoubtly the outstanding features. These remarks lead to an original analysis principle since parts of an ad text are not recognized.

An expert system approach has been adopted to design the interface and the next point deals with the knowledge required from experts. It concerns for the essential part the specification of what is an acceptable ad from both syntactical and semantic aspects. This is achieved by means of a two-level formalism founded on attribute context-free grammars. The use of this knowledge through an inference engine is also approached.

A first prototype has been implemented in 1983, on top of a Multics system. However, this context was not adequate, so that this interface has not been really experimented by large public users up to now. This state of things prevents from any effective evaluation of this new kind of interaction means. We are now in the process of implementing a second prototype, running on a dedicated SPS7 under Unix.

The last part of the paper deals with the extension of that kind of interface, called CALIN. The idea is to allow users to query information systems by means of queries which are as close as possible to ads. The principal reason of the generalization attempt is the natural simple aspect of such questions and the aid provided by the functions offered by the CALIN interface. In fact, we introduce the notion of objects which are semantic entities seen by the users. A given query refers to only one object and involves three parts : the object, the desired results and the selection expression close to an ad -from a structural viewpoint.

A prototype related to the real estate domain has been developed in cooperation with an ad expert. It validates the formalism and the adopted analysing policy. The generalization of the approach is an exciting trend since it could end up to an intermediate way for query formulation, and consequently, we intend to investigate this area in more detail.

BIBLIOGRAPHY

[BOSC 83] P. BOSC et al.
HAVANE : un système de mise en relation automatique de petites annonces.
INRIA report 223, July 1983.

[BOSC 85] P. BOSC et al.
A high-level user interface for classified advertisements management.
Proc Jahorina Congress, Sarajevo, April 1985.

[BROV 85] C. BROVEMAN, W. CROFT.
A knowledge-based approach to data management for intelligent user interfaces.
Proc 11th VLDB Conference, Stockholm, August 1985.

[BUBE 82] J. BUBENKO et al.
A declarative approach to conceptual modelling.
Information systems design methodologies : a comparative review. Olle, Sol, Verrijn-Stuart eds, North-Holland, 1982.

[CHAM 76] D.D. CHAMBERLIN et al.
SEQUEL2 : A unified approach to data definition manipulation and control.
IBM Journal of research and development, vol 20, n° 6, 1976.

[CHAU 83] A. CHAUFFAUT et al.
On the use of a database machine for fuzzy querying.
Proc 3rd International Workshop on Database Machines, Munich, September 1983.

[CHAU 85] A. CHAUFFAUT et al.
Extending a SEQUEL-like language to allow fuzzy querying.
Proc 1st International Fuzzy Sets Association, Palma de Majorqua, July 1985.

[CODD 78] E.F. CODD et al.
Rendez-vous version 1 : an experimental english-language query system for casual users of relational data base.
IBM Research Report RJ2144, January 1978.

[COUR 85] M. COURANT, S. ROBIN.
Classified advertisement analysis in the context of an expert system in ad matching.
Natural Language Understanding and Logic Programming, Dahl and St-Dizier eds, North-Holland, 1985.

[DERA 83] P. DERANSART.
Logical attribute grammars.
Proc IFIP Conference, Paris, September 1983.

[GREG 85] R. Mac GREGOR.
ARIEL - A semantic front-end to relational DBMS.
Proc 11th VLDB Conference, Stockholm, August 1985.

[HEIL 85] S. HEILER, A. ROSENTHAL.
G-WHIZ, A visual interface for the functional model with recursion.
Proc 11th VLDB Conference, Stockholm, August 1985.

[HEND 78] G.G. HENDRIX et al.
Developing a natural language interface to complex data.
ACM Transactions on Database Systems, vol 3, n° 2, 1978.

[KORT 84] F.H. KORTH et al.
System/U : A database system based on the universal relation assumption.
ACM Transactions on Database Systems, vol 9, n° 3, 1984.

[KNUT 71] D. KNUTH.
Semantics of context-free languages.
Mathematical System Theory, vol 5, n° 1, 1971.

[LAUR 82] J.L. LAURIERE.
Knowledge representation and use. Part one : expert systems.
Techniques and Science of Informatics, vol 1, n° 1, 1982.

[LUBO 85] P. LUBONSKI.
Natural language interface for a polish railway expert system.
Natural Language Understanding and Logic Programming, Dahl and St-Dizier eds, North-Holland, 1985.

[ULLM 83] J.D. ULLMAN.
Universal relation interfaces for database systems.
Proc IFIP Congress, Paris, 1983.

[WOOD 77] W.A. WOODS.
A personal view of natural language understanding.
ACM SIGART Newsletter, n° 61, 1977.

SOLVING GRAMMATICAL AMBIGUITIES WITHIN A SURFACE SYNTACTICAL PARSER FOR AUTOMATIC INDEXING

Catherine BERRUT et Patrick PALMER

Group "Systèmes Intelligents de Recherche d'Informations"
Laboratoire Génie Informatique de Grenoble (IMAG)
B.P. 68 - 38402 Saint Martin d'Hères Cédex
France

ABSTRACT

This paper describes linguistic tools specifically designed for performant automatic indexing of natural language texts. By performant indexing, we mean the ability of the system to extract noun phrases (considered as main conceptual frames regarding text content) without processing full syntactic analysis of sentences (surface analyzer) both with its ability in learning unknown words. The paper describes the overall principles of this parser, emphasizing the use of syntactic networks and precedence matrix to fulfil the above goals of reducing the analysis cost and infering new vocabularies without interrupting the indexing process.

1. Introduction

Numerous linguistic tools have been studied and experimented during the post decades, most of them trying to overcome the whole problem of parsing or understanding natural languages. It is a well known fact that the problem is so tremendously complex that no one has been able to solve it despite many interesting, but partial, results. On the other hand, linguistic process are of prime interest in Information Retrieval [SALT85], the main contributions being of course the enhancement of automatic indexing and many man-machine interfaces through natural languages. Is that to say that reputed costly and slow linguistic tools are of no practical interest in our domain? Our opinion is that a solution may be found considering the usually large amounts of data handled by Retrieval Systems and their particular needs of automatic indexing: we have to carefully adjust the final objectives of the linguistic analysis in this particular context, and define tailored methodologies and softwares to fulfil them.

About the final objectives there is a general agreement to consider noun groups as the good indexing unit, which could help to overcome classical precision problems related to simple key-words. Is it then possible to define performant tools to locate these syntagms in large full text databases? One have to be aware that even noun groups recognition implies facing the problem of syntactic ambiguities. It is then of prime importance to thoroughly define which noun group syntax will be recognized, and limitate somewhat our objectives to the most common and less ambiguous patterns. From this point of view it is important to note that ambiguity patterns strongly depend on the considered natural language; problems which need semantic control in some language may be entirely solved at a syntactic level in an other. After the definition of the noun-group syntax, one have to define performant methods to extract them from sentences and solve the remaining ambiguities. The surface analyzer presented in this paper tries to match these objectives. Surface analyzer means that we try not to go in deep syntactic structures of the sentences, but rather to quickly find anchor points which may denote the beginning of a noun group, and process recognition from these points. In fact we only extract a subset of the noun groups, which type is given by the syntax (see further).

An other very important feature of the parser is the context of large data bases and automatic indexing is its learning abilities; a classical limitation of analyzers is the problem of unknown words which always stop the process or switch it to interactive dialogues for external informations before resuming. From our point of view the ability of the system to learn unknown words is at least as important as its overall time performances and constitutes one of the major objectives of this study.

In this paper we emphasize the first step of the surface analysis: the morpho-syntactic analysis and the solving of grammatical ambiguities. We also relate these problems to automatic word learning.

2. Indexing through Conceptual Groups

The idea of indexing through noun phrases has been proposed a long time ago [LEGU82]. Enhancement of precision may be expected from higher level definition of text content derived from such syntagmatic structures. Practical application to automatic indexing, if of prime interest, nevertheless means to face very specific problems related to the particular context.

As stated in the introduction, the first one concerns the solving of ambiguities which are common while parsing natural language sentences. Due to the large amount of text to be processed in automatic indexing applications, this has to be done both very efficiently and without any human interaction. Since solving ambiguities is still an open problem in computational linguistics, we have to restrain

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

ourselves to ambigity schemas we have a good chance to effectively solve. Fortunately, this constraint (at least considering the French language) is not impossible to overcome, since the most common noun phrase patterns may be solved without too much difficulty. Moreover the underlying indexing method does not imply deep understanding of noun phrases [KERK84] because the final choice of indexing terms implies reference to a thesaurus (which can further help to correctly understand noun phrases). Hence recognition of noun phrases may tolerate some remaining ambiguities which can be solved at a later step.

Efficiency may be ensured in limitating the overall parsing process to the very portions of sentences which have good chances to contain noun phrases. This is what we mean while considering the definition of a surface analyzer. The second important problem is related to automatic learning of unknown words.Our opinion is that such a feature is as important as the preceding one: due to the constant insertion of new documents in the corpus, the basic vocabulary is evolving. One can not imaginean "automatic" indexing process which stop on every unknown word. Hence autometic indexing corresponds to very specific linguistic tools, with uncommon capabilities considering the classical natural language parsers one can find in the litterature.

The idea of Conceptual Group was developed and experienced in our automatic indexing process [KERK84] They mainly consist in normalized representation of previously extracted noun groups, in such a way that they can easily be transformed in indexing terms. The term itself "Conceptual Group" clearly expresses that the noun groups they are built from are considered as the backbone of semantic information, or concepts [LEGU82] A syntactic definition of Conceptual Groups (according to the French language) is given in appendix one.

As an example, the following noun group, found in a sentence:

"le prêt dans les bibliothèques" (the loan in libraries)

would be transformed in the Conceptual Group:

"prêt dans bibliothèque" (loan in library)

Structural transformations may also occur, such as the reordering of adjective lists related to a substantive:

"le beau ciel très clair" (the nice very clear sky)

will give the Conceptual Group:

"beau clair ciel" (blue clear sky)

As may be seen in these examples, the Conceptual Group is a canonical representation of the original noun group where empty words (the articles for example) are discorded and significant words are not inflexed (canonical or lemmatized items).

One may find more details about the transformation of Conceptual Groups into indexing terms in [KERK84] [KERK85] through a pattern matching procedure involving a knowledge base.

3. Overall presentation of the morpho-syntactic analysis

Let us first detail what our objectives exactly are considering grammatical ambiguities and automatic word learning.

3.1. Grammatical ambiguities

Many linguistic studies have already attempted to classify the main categories of grammatical ambiguities. B. Vauquois and N. Nedobejkine [VANE77] consider three main classes of ambiguities, each of them being also subdivised in three subcategories:

- shape ambiguity
  - sheer polysemy
  - class ambiguity
  - morphological ambiguity
- structural ambiguity
  - composed words structure
  - logical component structure
  - grammatical component structure
- anaphoric ambiguity
  - syntactical discontinuity
  - ellipsis
  - anaphora

Except for the composed word structure the two last categories are not related to noun groups analysis but rather to sentence analysis and understanding. We are thus mainly concerned by the shape ambiguities which are related to word grammatical properties.

examples : "aire" which means either "surface" or "eyrie of eagle" is a sheer polysemy;
"porte" which means either "door" or "to bear" is a class ambiguity
"noix" which means "walnut" (singular or plural) is a morphological ambiguity.

Given this first (and important) linguistic limitation which establishes that we do not try in any way to go through full syntactic analysis of sentences, we have to make clear that the parsing process will try to solve only the ambiguities within sentence fragments which may contain noun groups. These limitations lead to the definition of the surface analyzer (the syntactic analysis occurs only on sentence fragments suspected to hold noun groups) and of what we named "interesting ambiguities": ambiguities whose resolution may yield a noun group.

The morpho-syntactic analysis concerns both the recognition of single words' grammatical properties (morphological analysis) and the verification of structural associations of words allowed by the language (syntactic analysis). These two kinds of analysis are usually deeply interconnected while processing natural languages; hence the term morpho-syntactic analysis expresses the overall association of these two basic components.

3.2. Automatic word learning

In order to avoid interruptions in the search for noun groups due to unknown words we have to set up a process which could allow the parser to automatically resume its work when the problem arises. Since the analysis is mainly based on word grammatical properties one can limit the learning of a new word to the setting of its root and its lexical categories, which have to be stored in the general dictionary for further use. Thus word learning is basically morphological learning. At a procedural level, morpho-syntactic analysis and word learning are deeply interrelated since both processes needs each other during the parsing. The overall strategy for word learning is presented apart for more clarity.

4. The morphological analysis

Through the use of a dictionary, the morphological analysis gives the list of all the grammatical categories of each word in the text [PALM81]. We can obtain:

- either a single grammatical category (the word is recognized without any ambiguity);
- or several grammatical categories (the word is a homograph)

- or no grammatical category (the word is not known in the dictionary).

A default list of grammatical categories is given to each unknown word, therefrom we only have grammatical ambiguities caused by homographies or unknown words.

The most important tool of the morphological analysis is thus a dictionary which contains the grammatical categories of each known word. Each word is divided in stem and suffix; these stems are stored in the analysis dictionary, the suffixes in multiple small specific arrays [PALM81].

A list of about fifty grammatical categories has been manually defined which can be divided in two groups:

- **the closed classes**: these are the categories which contain a finite number of words, for example the articles, the pronouns.
- **the opened classes**: these are the categories which contain a hardly numerable list of words, such as nouns, adjectives, adverbs and verbs, which content may evolve as the corpus grows.

Inside these opened classes, the irregular words, which mainly come from the language's heritage, can be considered as a stable subset which we have stored in the dictionary. We also have stored the closed classes [GREV80]. All these manually stored words are specially marked in order to distinguish them from the learned words.

As we know that the dictionary contains all the closed classes and all their homographs [TLF71], each new word, that is to say each word which is unknown in the dictionary, is neither irregular nor member of a closed class: so it is either a noun, an adjective, a verb or an adverb. If it is a noun, an adjective or a verb, the word is regular: its different forms (feminine, plural) can be automatically deduced from standard inflexion suffixes; if it is an adverb, it must have the suffix "MENT" because all the other French adverbs have been already stored.

So on one hand, the morphological analysis gives for each known word a list of grammatical categories, and on the other hand (because of the difference between the closed and opened classes), we can resume the analysis when an unknown word is met by assigning to it a default list made of a subset of the four opened classes. The syntactic analysis may try further to select one of them [PALM86].

example: consider the sentence

il	*mange*	*des*	*bonbons*
he	*eats*	*some*	*sweets*

the morphological analysis gives:

PRPS	VBCJ	ARTI PREP	?	PNTF

where PRPS = PRonom PerSonnel (personal pronoun)
VBCJ = VerBe ConjuGué (inflected verb)
ARTI = ARTIcle
PREP = PREPosition
PNTF = PoiNT Final (full stop)
? = empty list for an unknown word of the dictionary. This empty list might be replaced during the syntactic analysis by a subset of the opened categories, here *common substantive, qualifying adjective, inflected verb*. The word can not be an adverb because it does not belong to the closed list of irregular adverbs, and does not have the standard suffix "MENT" of the opened list of adverbs.

5. The syntactic analysis

The syntactic analysis first applies a syntactic filter on the list of grammatical categories obtained from the morphological step. This filter is based on positional relations between grammatical categories of consecutive words in a sentence, which is synthetically defined by a so called "precedence matrix". If we consider the transition possibilities between grammatical categories of consecutive words as links, and categories as nodes, we can view the sentence as a (generally not linear) network. Non linearity arises when several links start from or converge to a single node: this corresponds also to ambiguity (see examples below). The filtering step first enables the syntactically correct links between nodes, but also discards unauthorized configurations (i.e. some of the grammatical categories of words may be discarded). At the end of this step some ambiguities may still remain: some words may still have multiple grammatical possibilities, so that several pathways may still be considered in the network. The syntactic analysis then goes through a second step to solve remaining "interesting ambiguities"; i.e. it tries to linearize fragments of the network which may lead to a noun group. This process involves the recognition of ambiguity patterns and the application of standard solution for each of them. At the end of this step the network is linearised on every portion which contains a noun-group (partly linearised network). Hence we have solved only "interesting ambiguities" without going into deep sentence analysis.

5.1. The precedence matrix

This first syntactic tool is based on the boolean precedence matrix of C. Fluhr [FLUH77], which indicates the possibility or unpossibility for two grammatical categories to occur in successive order within a sentence. For example noun - epithet adjective is allowed, article-verb is not allowed. Unlike C. Fluhr, our matrix gives five kinds of results for each couple of entries [PALM86]:

- blank when the sequence of the two categories has not already been seen;
- 0 when the sequence is forbidden, for example the sequence common substantive-definite article;
- 1 when the sequence is allowed without any condition, for example the sequence substantive-preposition;
- 2 when the sequence is allowed if there is a number concordance between the two categories, for example the sequence personal pronoun-inflected verb;
- 3 when the sequence is allowed if there is a number and gender concordance between the two categories, for example the sequence finite article-common substantive.

Therefrom, for each pair of consecutive words, we refer to the matrix, and the unable sequences simplify the combinational of the initial network. This step is very performant since it involves only dictionary accesses and matrix consultation. On the other hand the five valued elements of the matrix greatly improve the process, making full use of the inflexion properties of French (gender and number inflexions of adjectives, articles, past participles,...)

5.2. The use of the precedence matrix

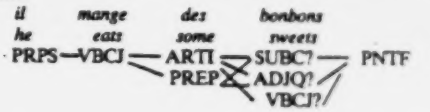
First a word may be unknown in the dictionary; in that case, we temporarily assign a subset of the opened classes in place of the empty grammatical list. This set is chosen among the four opened classes previously defined. Preselection may occur through suffix analysis, which are standard among each opened class. At the end of the syntactic analysis, the new word is stored in the dictionary

with the deduced category. Since there is no absolute certitude about the effective list of grammatical categories the new word may bear (we have only found a possible subset of them in a particular context), learned words are marked in the dictionary to avoid confusions with completely defined words, and allow further updatings of their categories. In the following examples question marks (?) hereforth indicate deduced categories of the unknown words, and stars (*) the learned words.

Through the use of the precedence matrix on the morphological data, we obtain a first network in which different possibilities may subsist:

- ambiguities due to homographies or unknown words;

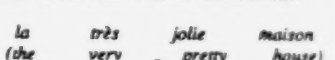
example: The example we had at the end of the morphology becomes



where SUBC = SUBstantif Commun (common substantive)
ADJQ = ADJectif Qualificatif (qualifying adjective)
VBCJ = VerBe ConJugué (inflected verb)

- missing of sequence between two words because of
  - either a lack of information in the matrix

example:

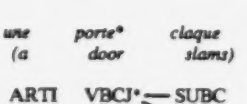


where ARTD = ARTicle Défini (definite article)
ADVB = ADVerBe (adverb)

If M is the precedence matrix,
if M(ADVB,ADJQ) = "blank", we have to put 1 in (ADVB,ADJQ) during the up-date of the matrix. Filling the precedence matrix content is part of the learning of basic syntactic rules of the natural language: one can assume that running analysis on enough manually solved texts will delete most of the blanks. This case therefore should be never encountered in current exploitation provided that extensive learning has been processed.

- or an incompletely learned word. At the beginning, the dictionary contains the closed classes and the irregularities. When an unknown word is found, it might be inserted with only one grammatical category, and if this word is a homograph, it may afterwards occur with another category which we don't know at the moment. When such a problem is detected, we must re-consider the word as an unknown word, and then if a new category is deduced, we insert this new category in the dictionary with the already deduced categories.

example :



where "porte" effectly means either "door" or "to bear", but was only learned as a verb during a previous analysis.

As M(ARTI,VBCJ) = 0 and "une" belongs to a closed category, its grammatical categories are perfectly known, "porte" must be reconsidered as an unknown word, and additional categories are affected according to its spelling and allowed transition with the preceding category (ARTI) and the following categories (SUBC,VBCJ), that is to say SUBC ADJQ.

Thanks to this resolution of the lack of information between two words, there is no more gap in the syntactic network but there may be remaining ambiguities. We have now to solve this combinatorial problem in order to linearise the network. The following examples summarizes the whole process on more complex cases.

example 1: consider the sentence

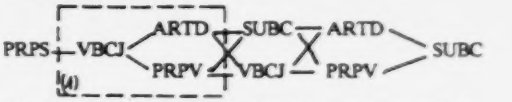
il ouvre les fenêtres le matin
(he opens the windows in the morning)

the morphological analysis gives us the following grammatical categories:

PRPS VBCJ ARTD SUBC ARTD SUBC
PRPV PRPV

where PRPV = PRonom PréVerbal (preverbal pronoun)

We might consider the following network before the syntactic filtering which exhibits the complete set of uncontrolled transition:



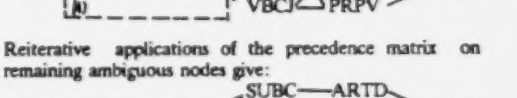
We can simplify this network with the precedence matrix:

Consider the subset of the matrix:

	VBCJ	PRPV	SUBC	ARTD
VBCJ	0	0	1	1
PRPV	1	1	0	0
SUBC	1	1	1	1
ARTD	0	0	3	0

We can read in the subset:
M(VBCJ,ARTD) = 1 (allowed)
and M(VBCJ,PRPV) = 0 (not allowed)
which allows us to solve the first ambiguity enclosed in the box (1) by discording the PRPV category for "les".

We obtain the simplified network:



Reiterative applications of the precedence matrix on remaining ambiguous nodes give:

In this case, the precedence matrix alone linearises the whole network; there is no remaining ambiguity.

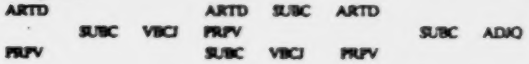
example 2: consider the sentence

le propriétaire vend la ferme le mois prochain

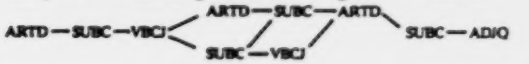
(the owner sells the farm next month)

where "ferme" is a French homograph meaning either "farm" or "to close".

The morphological analysis gives the following grammatical categories:



The precedence matrix gives the following filtered network:



We can see that the filtering is not always able to solve all the grammatical ambiguities.

5.3. The ambiguity patterns

When the syntactic filtering has been applied on the morphological output, the syntactic network may still be ambiguous. We have now to linearise the network locally by solving the "interesting" ambiguities which affect the components of the nominal syntagms. We use a set of ambiguity patterns whose solutions are known and named "solution schemas" [BERR85]: each time an ambiguity pattern is recognized in the network, we replace it by its solution in place of the ambiguous pattern.

Generally we can say that the filtered network is quite simple because of the small number of grammatical categories (about fifty) which constitute the linguistic model. Thereby we think that a limited number of ambiguity patterns will be sufficient to fulfil the objective of partial linearising;

5.3.1. Definition of ambiguity patterns

Ambiguity patterns are transformational rules made of:

- a left part which is an ambiguous pattern which has to be matched on the network. Since we are concerned by the solving of "interesting ambiguities", a common property of all these patterns is the presence of at least an ambiguous substantive in their definition (i.e. a homograph having the substantive category among others). This left part represents a non linear (i.e. ambiguous) network.
- a right side which is the solution of the ambiguity pattern, called the solution schema. It consists of a linear network which has to be substituted to the original non linear network matched by the ambiguity pattern. A common property of solution schemas is that they always exhibit noun groups structures, i.e. they always turn as a solution a syntagm which contains at least one substantive.

We thus have the general definition of an ambiguity pattern:

< *ambiguous pattern* > ---> < *solution schema* >

which may be read as "if the left part matches a sentence fragment, then apply the solution given by the right part".

We have now to define in more details how an ambiguous pattern is described and how a solution schema is applied on the network.

5.3.1.1. the ambiguous patterns

According to the surface analysis strategy, we want to perform deep analysis (and thus ambiguity solving) only on sentence fragments which could eventually contain noun groups [PABE85]. Thus we need special patterns which denote the beginning of potential noun groups, and trigger ambiguity patterns. Hence the name of trigger patterns we give to them. According to the French language and the syntax of noun groups, we give the following definition of these triggers:

TRIGGER PATTERN ::= ITEM0 ITEM2

ITEM0 ::= ITEM1 | CONJUNCTION ITEM1 | ITEM1 CONJUNCTION

ITEM1 ::= STRONG-PUNCTUATION | PREPOSITION | INFLEXED-VERB | INFLEXED-AUXILIARY | NEGATION-VERB

ITEM2 ::= ARTICLE | NON-QUALIFYING-ADJECTIVE | SUBSTANTIVE

STRONG-PUNCTUATION ::= PERIOD | SEMI-COLON | LEFT-PARENTHESIS | RIGHT-PARENTHESIS | QUESTION-MARK | EXCLAMATION-MARK | DOUBLE-PERIOD

Note: sequences ITEM0 - ITEM2 are restricted to those allowed by the precedence matrix.

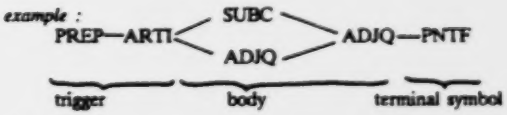
As a consequence, every ambiguous pattern has a trigger pattern as its leftmost components. Then we consider the body of the ambiguous pattern, which is a variable length sequence of items containing at least one substantive (possibly ambiguous) and various occurences of adjectives, articles and past participles (possibly ambiguous too). There is no general syntax to describe bodies; each of them is defined according to the main ambiguity case which has to be solved (see below some examples). The only general rule is that they constitute with ITEM2 as leftmost item, a superset of the noun groups' syntax.

The last component of the ambiguous pattern is the terminal symbol which is either a strong punctuation or a non-nominal item:

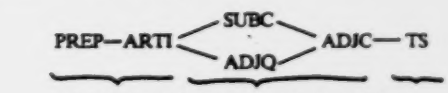
TERMINAL-SYMBOL ::= STRONG-PUNCTUATION | NON-NOMINAL-STEM

Thus the general definition of the ambiguous patterns may be summarized as follows:

AMBIGUOUS-PATTERN ::= TRIGGER-PATTERN BODY TERMINAL-SYMBOL



One can note that this definition may be somewhat generalized in the following way:



where TS stands for any terminal symbol.

This suggests the idea of reviewing somewhat the definition of grammatical categories in order to simplify the definition of patterns and to reduce their number by giving to each of them a broader scope. This also suggested by our final goal which is the extraction of noun groups: their

syntactic definition involves grammatical categories which must fit with the results of the transformational process undertaken by ambiguity patterns. This means that we are interested only in noun group recognition, and not in noun group deep structure which would be mandatory in natural language understanding or automatic translation: consider for example the pronouns, the precedence matrix considers several classes of them, such as personal pronouns, demonstrative pronouns,... while the syntax of noun groups only consider a global class of pronouns; It is also enough for us to consider "un" (one, a) as a determinant whether or not this item is an article or a cardinal adjective in a particular context, is not important since the noun groups' syntax only involves determinants as a whole. Thus why to bother with refined categories? There are two reasons for that: the first is that we plan to use the same linguistic tool for the analysis of natural language queries which is clearly concerned by refined syntactical structures, the second is that at the experimental step the balance is still not clear (considering time performances) between refined classification of grammatical categories, ambiguity solving and automatic learning of words. The system still has to be somewhat tuned in this way; our opinion is that only extensive experiments will allow us to adjust this balance properly through eventual simplifications of the precedence matrix.

5.3.1.2. Solution schemas

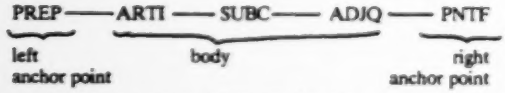
It consists of a linear network which will replace the whole non linear network described by the ambiguous pattern [BERR85]. In the same way as for ambiguous patterns, solution schemas are made of three parts:

- the left point: which is a unique item corresponding to ITEM0 of the trigger pattern;
- the body which is a linear network with ITEM2 of the trigger pattern on the leftmost item, bodies always correspond to noun groups;
- the right anchor point which is a unique item corresponding to the terminal symbol of the ambiguous pattern.

Thus the following general definition of solution schemas:

SOLUTION-SCHEMA ::= LEFT-ANCHOR-POINT
BODY RIGHT-ANCHOR-POINT
LEFT-ANCHOR-POINT ::= ITEM0
RIGHT-ANCHOR-POINT ::= TERMINAL-SYMBOL
BODY ::= ITEM2 NOUN-GROUP

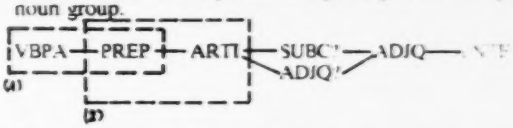
Anchor point delimit the scope of the transformation within the original network.
Example: this solution schema corresponds to the previous example of ambiguous pattern:



We now consider an example illustrating the whole process of applying ambiguity patterns.
Example:
Consider the sentence fragment:
"... mangé par le chat noir ."
(... eaten by the black cat ."

The morphological step and the syntactic filtering has given:
"... *mangé par le chat noir* ."
VBPA— PREP— ARTI—SUBC?– ADJQ— PNTF
ADJQ?

One can now imagine a window containing two consecutive words in position (1) and which quickly "slides" along the network checking for triggering configurations. In position (1), nothing happens since VBPA-PREP is not a trigger pattern (see above), then the window slides one word rightward and checks for PREP-ARTI (position (2)) which is a trigger configuration. Then every ambiguity pattern having this trigger configuration is activated because we detected the potential beginning of an ambiguous noun group.



The surface analysis now switches to ambiguity solving, which consists first in finding among the activated ambiguity patterns the one who matches the remaining items. This pattern matching process is supervised by a finite state automata which validates or unvalidates activated ambiguous patterns depending on the current network component being checked. When unvalidated an ambiguous pattern is disactivated and no longer considered in the process. When a terminal symbol is encountered (here PNTF), every activated ambiguous patterns but the one which positionaly corresponds to this final symbol are desactivated. Then the solution schema of the remaining ambiguity pattern is applied between its left and right anchor points.

Thus the nominal group "le chat noir" (the black cat) is extracted in the considered example, applying from position (2) of the window the ambiguity pattern described above.

Questions arise now about the effective construction of the set of ambiguity patterns which we are discussing hereafter.

5.3.2. Construction of the ambiguity patterns

It is of course difficult to predict which ambiguity cases will be found while processing texts; exhaustive lists of such cases can hardly be obtained from a strictly linguistic study. So intricate are the syntactic patterns that we found worthwhile to automatically extract them from experimental investigations on test texts. Since we have to define ambiguity patterns it sounded reasonable to investigate texts with "empty patterns" i.e. patterns defined as mentioned in 5321, but with an empty right part. This was a way to analyze which sentence fragments, ambiguous or not, could be further submitted to find ambiguity patterns. We were surprised to see that they were of them, at the moment less than 20, which is worthly due to the effectiveness of the syntactic filtering. We still have to process further investigations to completely ascertain this observation.

The second point concerns the overall consistency of the set of ambiguity patterns: what should happen if we defined concurrent patterns i.e. patterns with the same left part and different right parts? Or if there is one inclusion relation between them? We have to be careful at that or the system would not be deterministic (i.e. different results obtained depending on the first applied pattern).

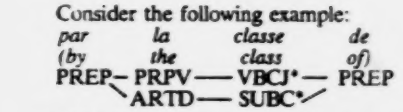
There are two rules which assure consistency of this set and which may be easily checked among it:

- no ambiguous pattern's body may be included in an other's one
- no partial covering between two patterns, except if they do not differently solve their common ambiguity.

Actually this is done by the definition we gave to the left part of the patterns.

5.3.3. The secondary filtering

Consider the following example:



This kind of "interesting ambiguity" may be solved by using the results of the precedence matrix. The sequence PREP-VBCJ is forbidden by the matrix, that is to say that after a preposition, we can not have an inflected verb. On the other hand, the sequence PREP-PRPV is allowed: "pour le voir" (to see him); and PRPV-VBCJ is also allowed: "je la porte" (I bring her).

This problem is related to a basic property of the precedence matrix: precedence rules establish a necessary condition for syntactic correctness of sentences, not a sufficient condition. Transitive constraints such as in the example above can not of course be directly expressed in a binary matrix. One has to check for additional properties, here the non-existance of PREP-VBCJ in the matrix. Thus the sequence PREP-PRPV-VBCJ, allowed by the matrix just like in our example, must be forbidden because we must understand it like the sequence of a preposition of a verbal syntagm.

Therefore we have to built a list of complementary rules of simplification which simulatean at least ternary filtering on the noun groups.

Consider our example, we write:
PREP - PRPS - VBCJ --->

In fact the activation of these rules is the same as the activation of the ambiguity patterns. The difference between these rules and the ambiguity patterns is that these new rules do not respect the definition of the ambiguity patterns (see above).

6. Conclusion

Significant improvements of automatic indexing may be obtained from linguistic analysis, providing that specific tools are designed with respect to the particular needs and objectives of the topic: solving grammatical ambiguities among noun groups and automatic word learning. The prototype softwares we are developping through Pascal and Lisp are not completed yet, nor integrated in the automatic indexing module we have developped before. Extensive experiments are under this way to refine the linguistic models and the ambiguity patterns to set with more accuracy the minimal informations we need to fulfil our objectives. It should be clear that most of the ambiguities may be solved through this purely syntactic approach; some might need semantic controls which are beyond our objectives at the moment though we could implement some of them referring the knowledge base currently used in our indexing method. Automatic learning of words, when related to the problem of solving ambiguities clearly has some procedural limits: if too many words belonging to opened classes are unknown (i.e. if there is a high probability that words belonging to a same potential noun group are unknown), the ambiguity patterns would lack the minimal information to solve the corresponding ambiguities. Hence the opened classes of the dictionary should be first initialised with a basic vocabulary which may be extracted from representative texts (the same which are used for other purposes such as the setting of the precedence matrix and the ambiguities patterns). Because closed classes are already stored within the dictionary one can consider this as the only linguistic investment we have to do for a particular corpus (the precedence matrix and the ambiguity patterns are general linguistic models).

APPENDIX 1

Appendix 1-a
First noun groups' syntax

GCP ::= GCP1 (R GCP1)
R ::= RELP | AUX | PROX
AUX ::= 'être'
PROX ::= déduite

Appendix 1-b
Current noun groups' syntax

GC ::= ARG RELP ARG | NOM ARG | ARG
ARG ::= NOM | VINF | NOM NOM
GQ ::= CPN | ADJ [GADJ] | GADJ
CPN ::= PDE ARG | PDE NOM GQ
GADJ ::= ADJ PREP VINF | ADJ PREP NOM ADJ
PREP ::= PDE | RELP
NOM ::= common substantive | proper substantive
ADJ ::= qualifying adjective | ordinal adjective | cardinal adjective | past participe
VINF ::= infinitive verb
PDE ::= prepositions : "de", "du", "des", "d' "
RELP ::= other prepositions

where ARG means ARGument
GQ means Qualifying Group
CPN means Noun ComPlement
GADJ means Adjectival Group
PREP means PREPosition.

Example : "le chien dans la forêt" (the dog in the forest)
becomes
"chien dans forêt" (dog in forest)

APPENDIX 2
LIST OF GRAMMATICAL CATEGORIES

SUBC SUBstantif Commun
SUBP SUBstantif Propre
ADJQ ADJectif Qualificatif
ADJC ADJectif Cardinal
ADJO ADJectif Ordinal
ADJP ADJectif Possessif
ADJR ADJectif Relatif
ADJD ADJectif Démonstratif
ADJI ADJectif Interrogatif
ADJE ADJectif Exclamatif
ARTD ARTicle Défini
ARTI ARTicle Indéfini
ARTC ARTicle Contracté
PRPS PRonom PerSonnel
PRPV PRonom PréVerbal
PRPC PRonom Personnel Complément
PADV PRonom ADverbial
PRPO PRonom POssessif
PRDM PRonom DéMonstratif
PRRL PRonom ReLatif
PRIN PRonom INterrogatif
PREP PREPosition
CONJ CONJonction
ADVB ADVerBe
XECJ auXiliaire Etre ConJugué
XEIF auXiliaire Etre à l'InFinitif
XEPA auXiliaire Etre au Participe pAssé
XEPR auXiliaire Etre au Participe pRésent
XACJ auXiliaire Avoir ConJugué

XAIF auXiliaire Avoir à l'InFinitif
XAPA auXiliaire Avoir au Participe pAssé
XAPR auXiliaire Avoir au Participe pRésent
VBCJ VerBe ConJugué
VBIF VerBe ConJugué
VBPA VerBe au Participe pRésent
VBPR VerBe au Participe pRésent
ADVN ADVerbe de négation Ne
ADVP ADVerbde de négation Pas
LOCP LOCution Prépositionnelle
LOCC LOCution Conjonctive
INTJ INTerJection
NOMB NOMBre
PNTF PoNcTuation Forte
PARE PAREnthèse
GUIL GUILlement
VIRG VIRGule
DPNT Double PoiNT
ABRV ABRéviation

APPENDIX 2
REFERENCES

[BERR85] BERRUT Catherine, "Résolution des ambiguïtés grammaticales. Une première approche dans le cadre d'un analyseur de surface de la langue naturelle.", rapport de DEA, Grenoble, France, July 1985.

[COUR77] COURTIN Jacques, "Algorithmes pour le traitement interactif des langues naturelles", thèse d'état, Grenoble, France, 1977.

[FLUH77] FLUHR Christian, "Algorithmes à apprentissage et traitement automatique des langues naturelles", thèse d'état, Paris, France, 1977.

[GREV80] GREVISSE Maurice, "Le bon usage du français", Paris, France, 1980.

[KERK84] KERKOUBA Dalila, "Indexation et propriétés structurelles de documents dans un système de recherche d'informations", thèse de docteur ingénieur, Grenoble, France, 1984.

[KERK85] KERKOUBA Dalila, "Indexation automatique et aspects structurels du texte", proc. RIAO, Grenoble, France, mars 19-21 1985.

[LEGU82] LE GUERN M., "Les descripteurs d'un système de documentation. Essai de définition", Colloque traitement des langues naturelles, Clermont-Ferrand, France, 1982.

[PALM81] PALMER Patrick, "Etude de l'organisation d'un dictionnaire pour l'analyse du français.", rapport de DEA, Grenoble, France, 1981.

[PALM86] PALMER Patrick, Rapport de recherche à paraître, Grenoble, France, 1986.

[PABE85] PALMER Patrick, BERRUT Catherine, "Etude d'un analyseur de surface de la langue naturelle pour un système de recherche documentaire", Proc. CAIS conference, Montreal, Canada, June 1985.

[SALT85] SALTON Gerard, " A note on Information Retrieval Models and Theories", Proc. RIAO85, Grenoble, France, March 1985.

[TLF71] TLF, dictionnaire des fréquences. CNRS. Nancy, France, 1971.

[VANE77] VAUQUOIS Bernard, NEDOBEJKINE N. "Classification des ambiguïtés", Table ronde du CNRS. France, 1977.

A Design of a Distributed Full Text Retrieval System

Patrick Martin
Ian A. Macleod
Brent Nordin

Department of Computing & Information Science
Queen's University
Kingston, Ontario K7L 3N6

ABSTRACT

This paper describes the design of a distributed information system for full text retrieval. The system is similar in functionality to STAIRS and is being developed on a network of PC's interconnected by PC Network. The implementation is built on a generalisation of the remote procedure call concept. Communications are based upon the recent CCITT X.400 standard. Examples are given of the design strategy for a subset of the STAIRS system.

1. INTRODUCTION

There are a number of circumstances in which a distributed retrieval system would be of use. In the simplest case where there is not enough space on any one machine to hold the information, a large data base can be distributed over a number of relatively small machines. Another situation is where it is administratively desirable to hold logical collections of information at a single node. For example, people might want to maintain information about their own personal papers locally but yet allow access to this material by others. Another use of distributed data bases is in providing levels of security in sensitive areas. Access to the information stored at a particular machine can be restricted by the distributed system software. A further consideration is where there is not a general availability of certain storage devices. For example, one node on a net might include an optical disk and it might be desirable to store parts of all documents on this disk.

In this paper we describe the capabilities and basic design of a distributed full text retrieval system which is being built on a network of small personal computers. Initially, rather than develop a new design, we are adapting a subset of an existing system, namely that of STAIRS [IBM81], to run across the network. STAIRS is a widely available commercial product distributed by IBM. It is not a state of the art system but is well known and understood and thus provides a useful model for a prototype.

The implementation environment is a network of IBM PC's linked together through PC NETWORK [IBM84]. The implementation is being developed within the context of the recently announced CCITT X.400 communications standard [CCIT84]. In the remainder of the paper, a brief description of the structure of STAIRS is given. The types of distributed data bases to be supported are also described. Finally we discuss the actual implementation strategy.

2. FILE ORGANISATION

A STAIRS document contains two types of information. One type is information for which an index can be constructed and corresponds to the main text of a document. The other type is ancillary information and corresponds to what might be called the attributes of the document, such as date of publication, number of pages and so on. These attributes tend to be numeric though need not be. These pieces of information are termed "formatted fields". Since a document rarely consists of a single piece of text, STAIRS provides the ability to name "paragraphs" within the text. Paragraphs correspond to sections of the document such as the abstract, the authors, references and so on. Thus a STAIRS document consists of a set of fields some of which are considered to be free text and others to be formatted information.

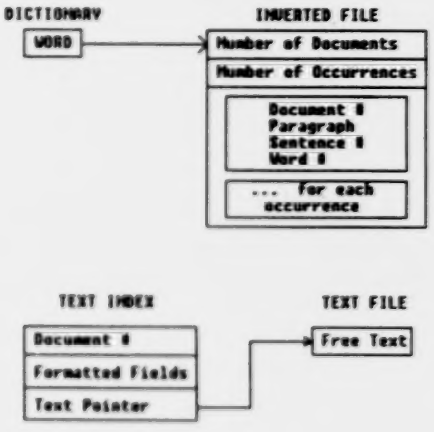


Figure 1 - STAIRS Structure

This work was partially supported by the Natural Science and Engineering Research Council of Canada under grants A0939 and A3308.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

Figure I illustrates the internal structure of STAIRS. The DICTIONARY contains a list of every unique word in the text. Associated with every word is an entry in the INVERTED FILE . This entry contains a count of the number of documents containing the word; a count of the total number of occurrences in the data base; and for each occurrence a specification as to where the word occurs in the text. This is given in terms of the document number, paragraph, sentence and position within the sentence. The TEXT INDEX contains the formatted fields together with a pointer to the location of the text which is contained within the TEXT FILE.

2.1. Network Considerations

A particular STAIRS data base might be configured in a number of different ways in a distributed system. There are at least three major possibilities and these are the ones we cater for in the current design.

(i) A data base might be completely contained within a single node. In this case, the purpose of the network will be simply to allow access to the node.

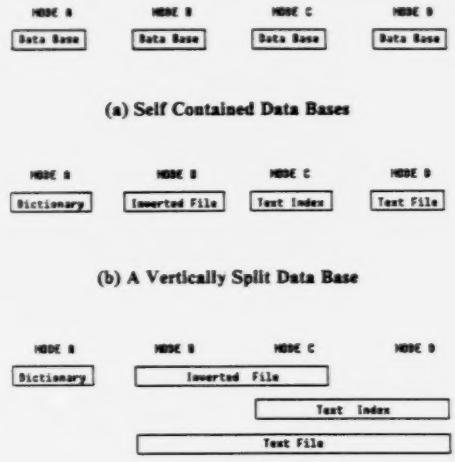
(ii) A data base might be split so that logical parts are at different nodes. We call this a *vertical* split. For example, an index might be at one node and the data at another node. Alternatively, the fixed format fields might be stored separately from the main text. Such splitting might be desirable for security, space or other reasons.

(iii) The data base might be fragmented so that sections are at different nodes. We call this a *horizontal* split. Horizontal splits would arise where the quantity of data was too large to store at a single node.

The three configurations are illustrated in Figure II. In (a), we illustrate two data bases each stored on separate nodes of a four node network. In (b) a STAIRS data base is split vertically with each of the four major components at a single node. Finally (c) shows the case where the inverted file is spread over two nodes, B and C, the text index over C and D while the text file is spread over B,C and D. We do not presently allow for combined vertical and horizontal splits. This is not for any fundamental reason other than the lack of any apparent need for such a degree of fragmentation. In the illustrations provided later, we will use the example of a horizontal split, this being the most general.

3. IMPLEMENTATION STRATEGY

Corresponding to the three major data structures, the dictionary plus inverted file, the text index and the text, STAIRS provides three separate retrieval operations. These are: SEARCH, for the dictionary file; SELECT for the text index; and BROWSE for the text. (A good introduction to these facilities is provided in [SALT83].) We are implementing these operations using a set of file handling procedures. In a distributed environment these procedures will, in general, be distributed across the nodes of the network. Therefore we will need the ability to have procedures residing at one node be available to users at other nodes. This capability is provided through *remote operation calls, (ROC's)*. These are an extension of the well known remote procedure call (RPC) facility [BIRR84,COOP85,LIN85]. RPC's are *transparent* in that they look to the user like a normal procedure call. This is not always appropriate. For example, data may be duplicated at several nodes and we may want to direct an operation to a particular node. Another case is where we might be able to improve efficiency by having independent operations run simultaneously on different nodes. Some form of concurrency control will be needed to facilitate the management of such



(a) Self Contained Data Bases

(b) A Vertically Split Data Base

(c) A Horizontally Split Data Base

Figure II - Example Organisations for a 4-Node Configuration

asynchronous operations.

For example, the STAIRS command:

SEARCH
1 INFORMATION AND RETRIEVAL

will involve looking up "INFORMATION" in the dictionary and retrieving the relevant entry from the inverted file. The same will be done for "RETRIEVAL" and the two results will then be "AND'ed" together. Obviously, there is no advantage in performing these two operations synchronously (or serially) and there are many reasons for not doing it. For example, there may be multiple copies of the dictionary on different nodes which can be accessed simultaneously. Or, if the dictionary and inverted file are on separate nodes, a second dictionary lookup can be started prior to the first inverted file retrieve being completed.

An ROC is basically an RPC with additional capabilities. These are:

(1) *Synchronous/asynchronous:* In the case of a synchronous call, the calling program waits until the call is completed. For an asynchronous call, the program continues immediately after the call is initiated. It must later include an explicit check to ensure the operation has in fact completed.

(2) *Directed/undirected:* Here the calling program may specify through a directed call which node is to process the request. A particular user may, for example, wish to keep a private copy of the dictionary at a particular node for efficiency reasons. It will then be necessary to be able to direct a lookup call to that specific node. For an undirected call, the ROC will be directed to any of the available nodes according to some previously determined policy.

(3) *Unicast/multicast:* For a unicast call only one of the possible nodes is accessed. In multicast, all of them are accessed. For example, if the dictionary was replicated

at several nodes, a "greedy" program might direct a lookup request to all of these nodes and use only the results returned by the first node to complete.

Remote operation calls are used by *clients*. A client program invokes an ROC to obtain *services*. A *server* is the program residing at a particular node that provides a service. Figure III illustrates the structure of our design. The ROC's of the application (client) program are used to generate *Procedure Stubs*. These in turn transmit information such as the name of the procedure being called and its parameters and package it for handling by the *Communications Protocol*. The protocol determines the destination of the ROC by consulting a *Directory Server*. In the following sections we discuss the design of the stubs, protocol and directory server.

3.1. Procedure Stubs

Procedure stubs are the interface between the program and the ROC system. The term *stub* is also used in the RPC system of Birrell and Nelson [BIRR84]. The stub for a particular program is automatically generated from the program source code by a preprocessor called the *stub generator*. There are two types of stubs - *send stubs* and *receive stubs*. A send stub is generated from the client (or calling) program and handles the ROC's of that program. A particular ROC in the program is actually implemented as a local call to a procedure of the same name in the send stub. The stub takes the parameters of the ROC and packages them for transmission to the server which in turn transmits the results back to the client. Transmission protocols are described in the next section.

Procedure stubs also control the error handling in the system. They set client accessible error condition variables should an error code be returned by the ROC mechanism, and call the appropriate optional client-supplied error routine. This routine is invoked if an error results from a remote call to that operation. If there is no error handling routine specified and an error occurs, the procedure stub forces a local error to simulate the error semantics of a failed local procedure call.

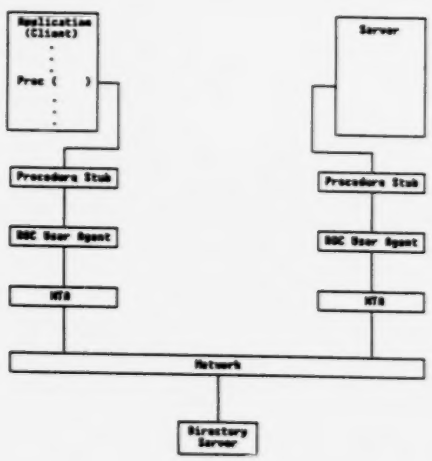


Figure III. ROC System Structure.

Programs that involve remote operations must contain special statements. These statements define operations provided by a service and how operations are used by a client. The statements have meaning to the stub generator but are treated as comments if encountered by the host language compiler. The ROC language extensions are preprocessed by the stub generator to produce code that can then be passed through a compiler. Preprocessing was chosen over altering the language and compiler because of the portability of a preprocessor and the relative ease of the solution. The language extensions are constructed such that a program can be compiled without first being preprocessed. Thus the preprocessor does not have to be run every time during the debugging of a host language program.

ROC "comments" are distinguished from regular comments by the addition of the string "ROC" immediately after the opening comment delimiter. The technique of using comment statements to hide language extensions has been used in many language implementations, for example Borland's Turbo Pascal [BORL83]. The stub generator produces the send and receive stubs by scanning the source programs and interpreting the special ROC statements in the programs into code for the stubs. Send stubs are generated for each operation "imported" by a client and receive stubs are generated for each operation "exported" by a service. ROC statements are illustrated in the examples of Section 4.

3.2. The Communications Protocol

The communications protocol is based on the CCITT recommendations on message handling [CCIT84]. These are contained in a series of documents known as the X.400 Series. These documents describe facilities to allow both people and machines to exchange messages. The X.400 model supports general, application independent message transfer and provides a set of well-defined protocols. Its use will also facilitate the porting of the ROC mechanism to different networks. The X.400 network model is shown in Figure IV. The *Message Transfer System* (*MTS*) is composed of a set of interacting *Message Transfer Agents* (*MTA*'s) which act as an interface between the network and an application. An application is a set of interacting *User Agents* (*UA*'s). The ROC system is viewed as a set of *UA*'s. The interface between the ROC system and an underlying network is the interface presented by the *MTS*. This is fixed for any underlying physical network. In the simplest case, when a client makes a remote operation call, control is transferred to the ROC mechanism resident at the client node. The stub takes the parameters of the ROC and packages them for the ROC User Agent. The ROC UA sends the ROC via the MTA to the appropriate server. The server's location is determined

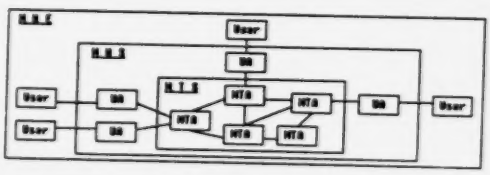


Figure IV. X.400 Network Model

through an interaction with a *directory server* as described in the next section. When the ROC completes, the send stub takes any values received from the ROC UA and passes both values and control back to the client program. The receive stub unpacks the parameters received from the UA and issues a local call to the appropriate operation. When the call completes, the receive stub packages the results and passes them to the UA for delivery to the client node.

3.3. Directory Server

The use of a separate directory service to store information about objects and their location allows a flexible binding of operation calls to actual nodes in a network [BIRR82, OPPE83]. The design of the directory service follows the draft recommendations outlined in a special report on directory systems by the CCITT Study Group VII [CCIT85]. System entities are identified by a *name*. The name is used to find an entry in a distributed database called the *directory*. Each entry in the directory contains a set of properties for the corresponding entity. The entities described in the directory can be single items or groups such as distribution lists. Operations will be available to query the directory; add and delete directory entries; add, delete and modify properties, and modify group memberships. A directory server is accessed via a remote operation call from a *ROC UA* at some node. The directory service can be provided by one or more servers and the information can be replicated in any manner.

The directory entries needed by the ROC mechanism are automatically created by the stub generator when the code for a server is installed at a node. Four classes of entities must be identified in the directory:

(1) *Service* - An entry for each available service in the system. The properties include a group containing the names of servers providing that service; a group containing the names of the operations provided by that service, and a group containing the names of any other services that are used by the service to implement its operations.

(2) *Server* - An entry for each server. The properties include the network address for that server, a description of the machine currently running the server; the name of the service that the server provides, and a group containing the names of any servers that the server can call upon to implement its operation. This final group of server names coincides with the group of service names in the corresponding service entry. Each service in the service name group has one or more of its servers' names in the group of server names for the entry. Information about the host machine can be used for any data representation conversions that may be required.

(3) *Operation* - An entry for each operation available from all services. The properties include the name of the service providing the operation, a group of parameter names, and a description of the operation. Each name in the paramater group points to a directory entry for the type of the corresponding parameter.

(4) *Parameter* - An entry for each parameter type. For simple parameter types such as integer, boolean, character or real, the properties include the data representations for the particular type on the host server machines. For user-defined types, the properties include the component type names.

The entry for a service is used by a *ROC UA* to identify the possible servers for an operation. Once the server's name has been found it can be used to access the appropriate server entry and find the address and any data representation information required. The operation and parameter entries are used to facilitate type checking by the stub generator and packing and unpacking of the call parameters by the send and receive stubs.

As an example consider the directory requirements of a distributed STAIRS system, using the horizontally split data base organisation of Figure II (c). We define STAIRS to be a service that employs four other services, namely a Dictionary Service, an Inverted File Service, a Text Index Service and a Text File Service. We assume there is a single instance of the STAIRS service at NODE A. The other services have servers as indicated in Figure II (c). The service entries for this STAIRS system are as follows:

```
STAIRS:       (SServerA,(Search,Browse,Select),
              (Dictionary,InvertedFile,
              TextIndex,TextFile))
Dictionary:   (DServerA,Lookup,())
InvertedFile: ((IFServerB,IFServerC),IFRetrieve,())
TextIndex:    ((TIServerC,TIServerD),TIRetrieve,())
TextFile:     ((TFServerB,TFServerC,TFServerD),
              TFRetrieve,())
```

The first property of each service is a group containing the names of servers that provide the service. The server names used in the actual directory have to be valid directory names. The names used above are for illustration and not necessarily the proper format. The second property is a group containing the names of the operations offered by that service. Again, these operation names will have to be valid directory entry names in the real system. The final property, if present is a group containing the names of other services used by the service being defined. Only the higher level STAIRS service uses other services. This use of other services is reflected by a nesting of ROC's in the operations of the STAIRS service.

The server entries in the directory for this example are as follows:

```
SServerA:     (NODE A,"AT,MS-DOS",STAIRS,
              (DServerA,IFServerB,IFServerC,
              TIServerC,TIServerD,TFServerB,
              TFServerC,TFServerD))
DServerA:     (NodeA,"AT,MS-DOS",Dictionary)
IFServerB:    (NodeB,"XT,MS-DOS",InvertedFile)
IFServerC:    (NodeC,"AT,MS-DOS",InvertedFile)
TIServerC:    (NodeC,"AT,MS-DOS",TextIndex)
TIServerD:    (NodeD,"AT,MS-DOS",TextIndex)
TFServerB:    (NodeB,"XT,MS-DOS,Text File)
TFServerC:    (NodeC,"AT,MS-DOS",TextFile)
TFServerD:    (NodeD,"AT,MS-DOS",TextFile)
```

The first property for each server entry is the network address of the node that contains that server. We let the node identifier in Figure II (c) be the network address. The remaining properties for the server entries are a description of the node running the server and the name of the service that the server provides.

The system also requires a directory entry for each operation provided by the services involved. The operation directory entries required are

---

```
Search:      (STAIRS,(String, Vector), _ )
Browse:      (STAIRS,(Vector, File), _ )
Select:      (STAIRS,(String, Vector, Vector), _ )
Lookup:      (Dictionary,(String, Integer), _ )
IFRetrieve:  (InvertedFile,(Integer, Vector), _ )
TIRetrieve:  (TextIndex,(Integer, File), _ )
TFRetrieve:  (TextFile,(Integer, File), _ )
```

The first property of an operation is the name of the service that provides that operation and the second property is a group of parameter names. The final set of directory entries for the STAIRS system is one entry for each parameter type used by the operations. All the parameter types used above are straightforward and their directory entries are not shown.

4. IMPLEMENTATION OF SERVICES

Our implementation is being developed using the C programming language [KERG78]. Each service is implemented as a single piece of software. The software design for the STAIRS service is shown here by means of algorithms written in a C-like notation. The algorithms are not chosen for their efficiency. Rather they serve to show the structure of a distributed information system implemented with our ROC mechanism.

When the software is installed on a node, the service is instantiated as a server and the appropriate information is recorded in the directory by the stub generator. A service may have multiple instantiations, or multiple servers. Each instantiation must have a unique name although the operation names defined are be the same for all the servers in the service. The directory binds the server names to the service name.

Each operation provided by a server is identified to the stub generator by an export statement. This statement defines the service provided by the server, the operation and its parameters. The parameter definitions indicate the type of each parameter and how each one is passed, that is by value (VAL) or by reference (REF). Parameters passed by reference require the ROC mechanism to return updated values for the parameters while parameters passed by value do not require any new values to be returned. For example the export statements for a STAIRS server are the following:

```
/*ROC export-remote service STAIRS operation
        Search (VAL string, REF vector) */
/*ROC export-remote service STAIRS operation
        Select (VAL string, VAL vector, REF vector) */
/*ROC export-remote service STAIRS operation
        Browse (VAL vector, REF file) */
```

A client program must identify all remote operations it intends to use with an import statement. The stub generator uses the import statement to index into the directory to find the entries corresponding to the matching export statements. This information is used for syntactic and semantic checking of the parameter lists of any calls to this operation.

Any client of the STAIRS service would contain import statements corresponding to the operations defined in the above export statements. Since the STAIRS service is also a client of the other three services in our system, the code for a STAIRS server contains the following statements:

```
/*ROC import-remote service Dictionary
        operation Lookup */
/*ROC import-remote service InvertedFile
        operation IFRetrieve */
/*ROC import-remote service TextIndex
        operation TIRetrieve */
/*ROC import-remote service TextFile
        operation TFRetrieve */
```

In our STAIRS subset, there are three high level operations, Search, Browse and Select. Additionally, and not shown here, we need a frontend which will parse STAIRS statements and invoke these operations. The Search operation searches the Dictionary (using the Lookup operation) for words that appear in the search parameter. For each word found, Search then uses the IFRetrieve operation of the Inverted File service to obtain the document numbers associated with each word. These document numbers are then returned as the result of the search.

```
Search (SearchParameters, DocumentNumbers)
string SearchParameters;
vector DocumentNumbers;
{
 integer LookupArray [array_size];
 vector RetrieveVector [maximum_documents];
 string Keywords [array_size];
 string Connectives [array_size];

 parse SearchParameters into separate lists of
   keywords and Boolean connectives;
 x = 0;
 for each search keyword
   {
    ASYNCH (Lookup (Keywords[x],
      LookupArray[x]), x);
    x++;
   }
 check to make sure all asynchronous calls
   have completed;
 x = 0;
 for each element in LookupArray
   {
    ASYNCH ( DIRECT (IFRetrieve (LookupArray[x],
      RetrieveVector[x]), "*"), x);
    x++;
   }
 check to make sure all asynchronous calls
   have completed;
 combine all the vectors in RetrieveVector
    using the Boolean connectives in Connectives;
 assign the vector created above to the return
    argument DocumentNumbers;
 return;
}
```

The Select operation, simplified for the purposes of illustration, searches the TextFile sequentially for particular values of a formatted field. An optional vector of document numbers restricts the search to particular documents. This is to handle the common situation where a Select is applied to the documents previously retrieved by a Search.

```
Select (SearchParameters, DocumentNumbersIn,
  DocumentNumbersOut)
string SearchParameters;
vector DocumentNumbersIn;
vector DocumentNumbersOut;
{
 if DocumentNumbersIn not empty
   {
    x = 0;  /* x is a document number */
    y = 0;  /* y is an index into the result vector
                 of document numbers */
    for the documents in DocumentNumbersIn
      {
       DIRECT (TIRetrieve(DocumentNumbersIn[x],
                 t1_file), "*");
       if the keyword expression SearchParameters
                 is in the temporary file 't1_file'
         {
          DocumentNumbersOut[y] = x;
          y++;
         }
       x++;
      }
   }
 else
   {
    x = 0;  /* x is a document number */
    y = 0;  /* y is an index into the result vector
                 of document numbers */
    for ALL documents in the text index
      {
       DIRECT (TIRetrieve (x, t1_file), "*");
       if the keyword expression SearchParameters
                 is in the temporary file 't1_file'
         {
          DocumentNumbersOut[y] = x;
          y++;
         }
       x++;
      }
   }
 return
}
```

The Browse operation takes as an input parameter a set of document numbers. It uses these to access the Text Index to retrieve the formatted fields. From the Text Index it accesses the corresponding entry in the Text File where the unformatted text is stored. This example is simplified. In practice the Browse operation can select particular parts of the document for display.

```
Browse (DocumentNumbers, DocumentTextFile)
vector DocumentNumbers;
file DocumentTextFile;
{
 integer t_pointer;
 file t1_file, t2_file;

 x = 0;
 for each document number in the vector
        'DocumentNumbers'
   {
    DIRECT (TIRetrieve (DocumentNumbers[x],
        t1_file), "*");
    extract text pointer 't_pointer' from t1_file
    DIRECT (TFRetrieve (t_pointer, t2_file), "*");
    DocumentTextFile = CONCATENATE (t1_file,
        t2_file);
    x++;
   }
 return;
}
```

The ROC's to the Text Index, Text File and the Inverted File servers are all implemented with the DIRECT function. A request to any one of these services is multicast to each server of that type and the results merged by the caller. This solution is acceptable in the case where there can be replication of the data among servers. If the data is partitioned among the servers then only a subset of the calls will actually apply. For the partitioned case we would require a system catalogue or dictionary to store information about the location of data [ALLE82]. The STAIRS servers could query this dictionary before formulating the ROC's.

The Search operation calls the Lookup operation using the ASYNCH function. This function invokes the operation asynchronously and allows Search to have all its dictionary searches running concurrently. Actual concurrency will be determined by the type and number of nodes running the Lookup operation. The example STAIRS data base only has a single Dictionary service so concurrency will only be achieved if the service allows for overlapped requests.

Search calls the Inverted File service retrieve operation using both DIRECT and ASYNCH. The use of DIRECT has been discussed above. Adding the ASYNCH function to the calls recognises the fact that the retrieve operations are all independent and can thus proceed simultaneously (subject to the limitations of the servers).

5. SUMMARY

Very little is required in the way of extensions to allow STAIRS to be used in a distributed environment, However certain policy decisions need to be made regarding attempts to access unavailable data and these decisions may vary according to the reason for the unavailability. Data might not be accessible because a node is down or for security reasons, for example. In the former case, it is probably desirable for the user to know as much as possible about the missing data while the opposite is true in the latter case.

We are currently developing the initial implementation. When this is completed we will then examine both these and other important issues, such as performance.

REFERENCES

[ALLE82] Allen F.W., Loomis M.E.S., Mannino M.V., "The Integrated Dictionary/Directory System", *ACM Computing Surveys*, Vol.14, pp.245-286, June 1982.

[BIRR82] Birrell A.D., Levin R., Needham R.M., Schroeder M.D., "Grapevine: An exercise in distributed computing", *Comm. of ACM* Vol.25, pp.260-274, April 1982.

[BIRR84] Birrell A.D., Nelson B.J., "Implementing Remote Procedure Calls", *ACM Trans. on Computer Systems* Vol.2, pp.39-59, Feb.1984.

[BORL83] Borland International, *Turbo Pascal Reference Manual*, Borland International, 1983.

[CCIT84] CCITT Study Group VII, *CCITT Recommendations on Message Handling*, Oct.1984.

[CCIT85] CCITT Study Group VII, *Draft Recommendations X.ds0 - X.ds6 (directory systems)*, May 1985.

[COOP84] Cooper E.C., "Replicated Procedure Call", *Proc Third Annual ACM Symp on Principles of Distributed Computing*, pp.220-242, Aug.1984.

[IBM81] IBM Corporation, *Storage and Information Retrieval System/ Conversational Monitor System*, Form no. SB11-5545-1, 1981.

[IBM84] IBM Corporation, *Technical Reference PC Network*, Personal Computer Hardware Reference Library, IBM Corporation, 1984.

[KERN78] Kernighan B.W., Ritchie D.M., *The C Programming Language*, Prentice-Hall Inc., EngleWood Cliffs NJ, 1978.

[LIN85] Lin K.J., Gannon J.D., "Atomic Remote Procedure Call", *IEEE Trans. on Software Eng.* VolSE-11, pp.1126-1135, Oct.1985.

[OPPE83] Oppen D.C., Dalal Y.K., "The Clearinghouse: A decentralized agent for locating named objects in a distributed environment", *ACM Trans. on Office Information Systems* Vol.1, pp.230-253, July 1983.

[SALT83] Salton G., McGill M.J., *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.

A common architecture for different text processing techniques in an information retrieval environment

G. Thurmair
SIEMENS AG
Munich

Abstract

The following paper gives an overview on a text processing software called REALIST (Retrieval Aids by Linguistics and Statistics) which integrates different text processing techniques into a common surface. It supports the user by offering the environment of a given term, using morphological, syntactic and statistic means. The user can call up the processing results, use it for indexing, classification or retrieval purposes and combine them as he wishes e.g. to set up a search logic. The text processing is done on a main frame computer, the results are transferred to a minicomputer where the evaluation is performed. REALIST is a stand alone package, fitting any existing search systems.

In the retrieval context, this technique reduces connecting time and improves the search results.

REALIST is able to run on English and German texts. Each REALIST component has been separately tested with good success. An integrated version is currently under test at the US Patent ad Trademark Office using 150000 english patent abstracts, and a german version is being tested with 12000 legal texts of the European Community.

1. Goals of the system

The analysis of the content of computer-stored textual material is a critical point for the efficiency and acceptance of automated systems, whether in office environments, in document retrieval systems, in full text retrieval or in electronic publication. Support of existing systems hardly goes beyond providing lists of all terms known to the system and some kinds of mechanic operations like truncation or adjacency; support of laboratory prototypes hardly goes beyond processing some hundred documents, and mostly cannot be given for real world files containing some hundred thousands of documents. REALIST gives higher level support on the basis of large amounts of data.

An analysis of requirements in the text database field shows the need to make the data more transparent to the user. The user should be offered as much information as possible for his information problem: He should know which terms are available in the file, which terms are related to them and in which environment they occur. At the same time, the irrelevant information like uppercase/lowercase letters, different spellings of words etc. should be hidden from him, because he thinks in concepts and not in strings. This holds in the field of search question formulation where a precise search should be performed in order to reduce cost-intensive log-in times and avoid undesired results; it also holds in the field of indexing where the content of a document should be described using appropriate terms which have to be given consistent with other documents and therefore to be examined carefully.

For these purposes, REALIST aims at an integration of different document analysis techniques which focus on different aspects of analysis. These are:

- Methods for the normalisation of textual forms with regard to spelling and inflection: The searcher should not have to worry about plurals, capitalisation at the beginning of a sentence and the like. These techniques also reduce and compress the system vocabulary (by 20 to 35 % depending on the langauge).

- Methods which are word-related and attempt to group words with common word stems. This is done not on string level (more or less sophisticated truncation operations) but on linguistic grounds using knowledge about word structures and word formation processes. With these techniques, a word environment is given for each concept.

- Methods which automatically provide syntactic relationships among terms and make a user independent of how a term concept is composed as a multiple term phrase: The user wants to be able to search for such expressions in terms of concepts rather than in mechanical adjacency

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

etc. operations. For each concept, a specification environment is given, showing how the concept is syntactically related to other concepts.

- Methods which present the conceptual environment of terms by means of statistical techniques; they permit the use of terms other than those originally chosen by offering somehow "semantically" related terms which are based on information how these terms occur in a specific given file. So, a correlation environment is given.

Each of these methods has been developed independently and tested on large real world files (WES84, SWA85, NIE84). Each of them has already led to significant improvements of retrieval results not only in laboratory tests but also in real world applications. It is to be expected that their integration and combination will result in a retrieval component of high functional capacity.

Other techniques - especially semantically based - are not far enough developed to permit successfull application to larger files. Such methods should, however, be given consideration in the system design; the same holds for all kinds of manually produced information (e.g. thesaurus structures).

2. System architecture

The system prototype has an architecture that permits independent components to be modularly integrated into a common retrieval "surface". Addition or removal of components is feasible without major effort and without affecting the other components.

This means that the content analysis of documents is organised in such a way that each part can operate largely autonomously, and that it supplies its result to an active central file which permits the integration and manages the knowledge of all the information available. This is accomplished by means of a "knowledge file".

The knowledge file is organised as a semantic network with terms as nodes and different relations (like "word environment", "specification environment", "correlation environment" and others) as links. The links are produced automatically by the different text processing components and added to the network independently. This garantees that components can easily be added or removed, simply by adding and removing network links. If other relations are available (like "broader term", "narrower term", "see also" in an existing thesaurus or automatically produced semantic information), just a new link is created and added to the knowledge file.

The word standardisation component is shared by all the other ones in order to get rid of irrelevant information; the standard citation forms are used as basic entries of the knowledge file. A link to the textforms of a citation form, however, is needed, because the textforms are the real terms to search with in a given retrieval system.

Each component then runs independently and builds up its specific links in the knowledge file network. The components currently used are
- MARS for the word environment
- SINSYN for the specification environment
- PFAU for the correltation environment.

They are described in more detail below. They produce their results and announce the information of what is available to the knowledge file. The production runs are performed on a mainframe computer with respect to the capacities needed.

Retrieval and evaluation of the REALIST information is done on a microprocessor. The processing results are transferred to the micro and a retrieval possibility is set up there. The user can evaluate the semantic network by calling up terms and information available to a given term. He can evaluate links of different types and mix them. He can enter the system with either a knowledge file term or a textform or a word stem and obtain the information he needs. He can store and edit terms and other information in a special user area and e.g. build up a search query. The retrieval uses windowing techniques and a mouse; the user just picks the terms he wants.

REALIST offers its knowledge to the user rather than using its knowledge automatically. The user keeps the initiative and can use the system's information according to his own needs. This is necessary as information needs are different and up to now, no valid and realistic automatic modelling of them is available.

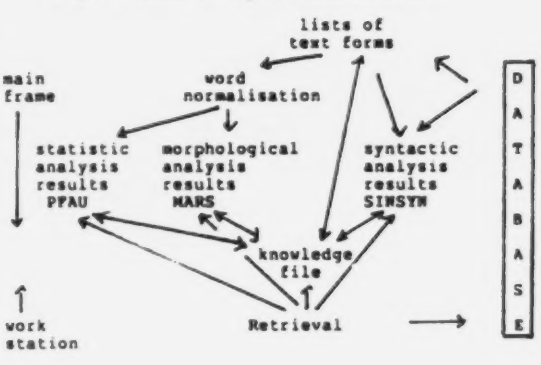
The overall architecture of the text processing components is given in Fig. 1.

3. Text processing components

3.1 Word normalisation

The first step of analysis involves the normalisation of the text words. This includes
- normalisation of spelling by appropriate conversion procedures
- the reduction of inflected text forms to non-inflected citation forms ("called", "calls", "calling" reduced to "call")
- the exclusion of words which have merely "grammatical meaning" (such as "at", "and",

Fig. 1: REALIST System Architecture



"would" etc.). These words do not enter the knowledge file, but stay in the textword files: they are searchable but no additional information is given.

Whereas the normalisation of spelling and the handling of stopwords can be accomplished with standard techniques, the reduction of text forms to citation forms is a bigger problem. This task requires the application of linguistic knowledge. At the same time, large amounts of data can only be analysed if the system operates fast, powerfully, robustly, and independently of type of text; this means that the analysis must be conducted independently of dictionaries, i.e. by application of rules.

The system developed for that purpose uses information on the handling of word endings and inflectional rules. These rules are partly independent of the word stems ("XXX-isations" always has to be reduced to "XXX-ation"), partly they are not ("kisses - kiss" but "busses - bus"). The information of rules and exceptions is given to the system by means of a file of examples; the system applies a learning technique to this file in order to generate the analysis rules. So the system can analyse every word independent of a dictionary. If an error occurs, it can be removed by just adding the correct handling of the wrongly analysed word to the learning file.

Since the rules are kept separate from the programs, the system can operate in different languages; the present version works with English and German textual material. The language of a given text is recognised by the system. The learning files containing the examples comprises about 3000 words per word class in English and about 3000-4500 in German. The system processes about 2000 words per second at an error rate of 3% for Enghlish and about 5% for German. It has been tested with several 100000 words of both languages.

Ths system is described in more detail in THU86. It includes components for the
- segmentation of an input text into sentences and words
- recognition of the language of text
- elimination of words of only grammatical meaning
- determination of the word classes of the content words
- handling of word endings
- generation of the citation form
- various sort and display functions.

This system of normalising textual forms produces standardised terms and at the same time a system of references indicating how and how often these terms occur in the original vocabulary. It forms the basis of the further analytic runs. As such, it offers advantages not only for retrieval, but also reduces the effort required for analysis since the vocabulary is reduced (by about 15 to 30 % depending on the language).

3.2 Word environment to a given term

This analysis component determines word stems within words and thus permits recognition of connections among differing words. This means that even different manifestations of words, like "electric", "electricity", "electrically-heated", "unelectrical", can be traced back to each other if they have the same word stem. Thus the user can search for terms without having to know in which manifestation they occur in the texts. Search can thus be conducted more accurately and more comprehensively than by using methods of truncation. The system not only avoids truncation errors ("selfish" as a hit for "$fish"), it also combines the effects of right, left and middle truncation without additional effort for the kernel information retrieval system.

The basis of this component is the system MARS, described in NIE84, THU84. It uses morphological techniques - a morpheme dictionary and a morpheme grammar in order to decompose words, to extract the stem(s) and to group together all words with the same stem. The system works with English (NIE84), German (THU84) and Spanish (MEY83) word material. The kernel is laguage-independent; language-specific information is put on data files. The dictionaries contain 7000-9000 word stems for each language. The system decomposes about 120 words per second, using an bredth-first left-to-right bottom-up parsing technique and a finite state analysis grammar. Ambiguous decomposition results are handled by applying special heuristics to extract the most plausible word structure. The system decomposes about 120 words per second with an overall error rate of 6%.

The determined word stems are inverted. If the user then searches with any particular term, he obtains all the words, compounds etc. that have the same stem as the search term, and he can select those he wishes to use for his search. Tests of this particular component in a real world retrieval environment have shown a considerable increase in the number of retrieved relevant documents without mentionably reducing precision. This holds for both free-text and descriptor systems (NIE84).

The result of this analytic component is a system of references which interrelates all morphologically related words. It gives the word environment of a given term.

Fig. 2: Example of MARS text processing
word environment of "absorb"

439	absorption	50	absorbed
36	absorbance	16	Absorption
15	absorb	12	malabsorbers
12	absorptions	11	absorbent
10	absorbing	10	malabsorption
7	absorbers	4	digestion-absorption
4	Absorbance	3	water-absorbing
3	absorber	2	absorbs
2	absorbant	2	non-absorbent
2	absorption-type	1	absorptivity
1	absorptively	1	noise-absorbing
1	non-absorbable	1	SO2-absorption

(frequency-term pairs, from FSTA; no word normalisation done here)

3.3 Syntactic techniques for Specification relations

Retrieval systems operate with "syntactical" aids when attempting to find term concepts composed of several words: Context operators are offered which calculate distances within which search terms may appear. Such procedures work mechanically and non-automatically, and complicated search logics must be constructed.

An optimisation of such techniques can be achieved by using linguistic techniques: Syntactic analysis gives the structural dependencies of words in a given phrase, not just the mechanic distance: "analysis of milk based cheese products" would produce a hit when searching with "analysis of milk", which is incorrect. Multiple-word expressions can be recognised via appropriate structuring and matching techniques. On a syntactical basis, any words which modify any other words in a given text can be found.

The syntactical analysis involves analysing the document texts and mapping them onto a normalised standard structure which is independent of the actual position of the word within the text (as later on in the search request). These structure reflects the syntactical position of the words relative to each other: E.g., expressions such as "egg white" will be closely correlated to "white of egg", "white of raw US-imported egg" etc., since the structural relation is recognised; in contrast, expressions containing "white egg" will rank much lower since the same terms appear, however in a different syntactical relationship.

The syntactic analysis recognises possible phrases using a robust, fast and surface oriented parsing technique. It is able to compare phrases that are structured differently; so the level of abstraction cannot be a purely syntactic one (like "attribute" or "prepositional phrase") but has to be a more abstract one, namely the specification relation: The analysis finds which terms are specified by which other ones, no matter how the specification is expressed on the syntactic surface. The structures which are produced are normalised in terms of specification relations, and the match between search query and document is done via these normalised structures. It is not necessary to restrict the analysis to noun groups of only two elements or to the maximum size of phrases. E.g., the user searching for "white of raw egg" would find "white of raw US-imported egg" in the best ranking class, and he would find "egg white" and "raw eggs" in a lower ranking class.

A prototype of this analytic technique which constitutes an integral part of the COPSY System (Context Operator Syntax, see SWA85) has been tested on 200000 English titles and index phrases (from FSTA and INSPEC) and on 20000 abstracts (from FSTA and US-Patent Office); the production of normalised structures took 500 CPU seconds for 15000 titles. The structures were correct in 95% of the cases. COPSY is one of the first syntactic analysis techniques that can process such large files at all.

A spin-off of this technique is part of REALIST:

Fig.3: Example of SINSYN text processing specification environment for "cheese"

specifying words		specified words	
11%	Cheddar	13%	manufacture
5%	Dutch	7%	production
5%	Emmental	6%	property
5%	processed	4%	ripening
3%	Ras	3%	acid
3%	Cottage	3%	component
3%	Mozzarella	3%	evaluation
1%	round	1%	transport
1%	Tilsit	1%	whey

(correlation - term - pairs; from FSTA)

As REALIST is term based, the COPSY structures are used to produce statistics of the vocabulary in question: For each word, all the terms that specify it syntactically, and all the terms that are specified by it, are given. The user can have an overview in which specification environment a term occures. This component, called SINSYN (see SWA86), operates on the basis of the COPSY syntactic structures.

The result of this analytic component is information as to which terms cooccur syntactically with any given word or phrase, and which terms specify which other terms.

3.4 Contextual term references via Statistical Extensional Relationships

This approach attempts to determine the statistical relationships within and among search vocabularies, as they actually exist in stored searchable material. The system supports the searcher in selecting appropriate search words by offering all those terms which are statistically correlated to a given term. The problem of the non-transparency of the search vocabularies is thus considerably diminished: "Maps" of the search vocabulary are provided which relate the differing words to each other and to other document parts, categories etc. Thus the actual relationships as they exist in a given search vocabulary are reflected.

This method, called PFAU (Programm fuer Ausdehnungstabellen, see WES84), operates as follows: In a first step, the search vocabulary is extracted from files available with a given information retrieval system (such as STAIRS, BRS etc.); PFAU is both database and retrieval system independent. Then extensional tables and listings are produced covering these files. For this purpose, the correlations existing among the descriptors of one document and those of different documents are calculated; in this manner extensional relationships of terms are determined. The results of these calculations indicate for any given descriptor its correlation with other descriptors in given document categories.

The PFAU component does not specify how (semantically) the terms are connected, nor does it automatically group together terms with respect to some kind of relevancy. It just offers the user additional terms he could use for his search

Fig. 3: Example of PFAU text processing correlation environment for "locate"

```
 26 documents
544 correlating descriptors

38% position          15% close
30% second            15% connect
30% space             15% front
27% extend            15% open
27% plurality         15% section
27% surface           11% chamber
23% apparatus         11% continuous
23% dispose           11% door
23% element           11% downward
23% side              11% horizontal
23% support           11% path
19% axis              11% permit
19% contact           11% place
15% adjacent          11% point
15% arrange           11% reduce

(correlation strength - term - pairs; from
a 300 patent document sample)
```

without prescribing him which ones to pick. So the initiative and any decision are still on users side. In this respect, PFAU differs from many other statistical approaches; the reason is that there is not enough knowledge about the behaviour of real world searchers, and therefore we do not try to model it automatically.

The production of PFAU results requires an analysis of a representative sample of documents of the given file and comparisons of the associated descriptor sets. To this extent the effort is dependent on the number of documents and the number of descriptors per document. Test have shown that about 2.5 CPU hours are required for files with an average of 50 words per document (after stop word lists), a search vocabulary of 20000 descriptors, and 10000 documents. With respect to full text retrieval, we found that the PFAU component should be restricted to document parts; tests have shown that the purpose (to show the extensions of given terms) can be reached by processing e.g. just the abstract of a text, as the result holds also for the full text.

The extensional tables have meanwhile been produced for various application of English and German files (MEDITEC, FSTA, Psychological Abstracts etc.) and have led to significant improvements of the retrieval results (see WES84). Their integration into the REALIST system (especially the word normalisation component) will improve their use still further.

3.5 The organisation of the analysis part: the knowledge file

The results of the analytic techniques are various, nonintegrated collections of information. All the techniques can run stand-alone; they concentrate on different levels of analysis (words - phrases - documents) and use different knowledge sources (morphology - syntax - statistics).

However, since they use common resources (e.g. the list of searchable words available in the files), it is useful to combine these information collections for a common retrieval and to use them as a kind of complex "thesaurus" in which the individual techniques contribute knowledge as to

- whether extensional tables are available for a given text word
- whether and how it appears in syntactic phrases
- which word stem it contains
- in which textual forms it can be searched
- in which document categories it appears and how often.

All this information is automatically constructed through the analytic results of the individual components; it forms the basis for the utilisation of the knowledge available in the system. The knowledge file itself can be seen as a semantic network whose nodes are the words and whose links are the different relationships. Other relationships, like manually built thesaurus relations, "see also"-pointers or AI-based semantic relations, can easily be added to the network.

Technically, the different analytic results are stored in independent tables and/or files which are interrelated and connected via the knowledge file. All terms are addressed by pointers to the knowledge file; these pointers are evaluated when information to a given term is required.

In order to store all the processing results, REALIST requires a certain amount of external storage. Several parameters can be used to control the storage requirement according to the specific needs of a user; at present, a version for a knowledge file of 100000 entries (representing about 4.2 million textform tokens and about 85000 documents of 100 words each) needs about 38 MByte storage. This can be provided on microcomputers; an alternative will be to use compact disks.

4 Retrieval

The retrieval part provides a set of functions operating on the various sources of information. These are accessible in an integrated fashion: For a search term of interest, the user can display the environments of this term by pointing to it with the mouse; the different environments are displayed in different windows. From these, the user can pick all the terms he is interested in an combine them in any way he likes.

E.g., he can enter the system with a search term. He can look it up in the knowledge file to see what information is available. He might call up the correlation environment to find other terms "semantically" related to his original one; he might pick some of them and store them in the user area. For some he might look up the specification environment in order to see whether or not the term is suitable for him; he might also find new terms. After having found the right concepts, the user can look up the word environments to find the best words for his terms; and finally he can evaluate the text-form relationships to obtain the strings the system can search with.

The REALIST retrieval is a tool which supports the user in formulating an exact search query or in classifying documents etc; it does no retrieval of its own, nor does it apply automatic means in answering a search request "better than" the user. This means that a user working memory has to be provided where the user can collect terms he is interested in, and that this memory has to be able to be edited. E.g. the user can pick his terms and than edit them in order to obtain a well-formed search query. A connection to the real search system (like BRS, GOLEM etc.) is not part of the REALIST software but could easily be added.

The retrieval operations within REALIST consist basically in evaluating the links of the semantic net. This can be accomplished via a uniform set of functions applied to the different relations. The following operations are possible:

- direct access to the knowledge file, to the text words and to the word stem file
- evaluation of the constructed relationships and retrieval of concatenated terms; sorting, marking and combining of elements of these concatenations
- manipulation of the user area: sorting, marking and deleting of entries, saving them in external files, editing the area etc.
- paging, marking etc. on all levels.

The retrieval component is to be part of the basic software for a workstation for information processing. It is therefore implemented not on mainframe computers (on which the REALIST text processing components are executed) but on smaller machines. It runs in a UNIX environment and is written in C.

5 Further development

REALIST is going to be tested at the US-Patent and Trademark Office. Questions of interest are:
- how much is the improvement in terms of recall and precision over a standard IRS?
- is it possible to have better results with a simpler IRS (e.g. without truncation and adjacency) plus REALIST?
- how can the components of REALIST interact? what could be improved?
- what kind of user reactions are available?

Results are expected for spring 87.

On the other hand, direct connections of the REALIST tool and given IR systems are feasible in order to set up a powerful workstation environment for text processing. Another application area for REALIST can be electronic publishing; a German REALIST version has been set up in this domain. Also, office environments need more powerful tools for search than just file names or string search.

Anyway, the distribution of IR functions between workstation and host have to be thought over; REALIST is a tool where these problems could be studied with.

Acknowledgements

The work described here was done by
- I.Buettel, M.Meya, G.Niedermair, M.Streit, G.Thurmair (morphology)
- A.Baumer, F.Schueler, Ch.Schwarz (syntax)
- E.Menth, A.Wessel (statistics)
- R.Krohn (retrieval).

Literature

MEY83 M.Meya: Treatment of Suffixes in Automatic Morphological Analysis. In: Linguistica Computazionale III, 1983, Supplement: L.Cignoni, C.Peters, Hsg.: Computers in Literary and Linguistic Research. (Proc. VII ALLC Symposion, Pisa), S. 199-206

MEY83 M.Meya: Analisis morfologico automatico del espagnol. in: Procesamiento del Lenguaje, Boletin 1, 1983, San Sebastian Fac. Informatica, 18-23

NIE84 G.Niedermair, I.Büttel, G.Thurmair: MARS - a Retrieval Tool on the Basis of Morphological Analysis. In: C.J.van Rijsbergen, Hsg.: Research and Development in Information Retrieval, S.369-382 (Proc. of the 3rd joint BCS and ACM Symposium, Cambridge, July 1984)

SWA84 Ch.Schwarz: Linguistische Hilfsmittel beim Information Retrieval. in: Nachrichten für Dokumentation 35, 1984, S.179-182

SWA85 Ch.Schwarz: Textrecherche mit Mehrwortbegriffen, erscheint in: Pro- ceedings der Jahrestagung der GLDV Hannover 1985

SWA86, THU86 Ch.Schwarz, G.Thurmair,eds.: Informationslinguistische Texterschließung. to appear Hildesheim 1986

THU83 G.Thurmair: An Integrated Algorithm for morphological Decomposition. In: Linguistica Computazionale III, 1983, Supplement: L.Cignoni, C.Peters, Hsg.: Computers in Literary and Linguistic Research. (Proc. VII ALLC Symposion, Pisa), S.269-277

THU84 G.Thurmair: Linguistic Problems in multilingual morphological decomposition. In: Proc. of the 10th International COLING Conference, Stanford 1984, S.174-77

WES84 A.Wessel: Retrievalunterstützung durch Ausdehnungsrelationen (Projekt Informationsbrücke). Frankfurt 1984

WES83 A.Wessel: Progress Report on Project Information Bridge. in: Proc. 6th Annual Intern. ACM SIGIR Conf., SIGIR Forum 17/4, 1983

COREL - A Conceptual Retrieval System

M. Kathryn Di Benigno*
Computer Science Department
Louisiana State University
Baton Rouge, LA 70803-4020

George R. Cross
Computer Science Department
Washington State University
Pullman, WA 99164-1210

Cary G. deBessonet
Law and AI Project
Louisiana State Law Institute
LSU Law Center
University Station
Baton Rouge, LA 70803-1016

Abstract

COREL is an experimental retrieval system that employs techniques of artificial intelligence. Articles of the Civil Code of Louisiana have been *conceptually* indexed using frame-based knowledge structures in hope of improving accessibility over traditional key-word retrieval systems. A set of macro packages has been developed to allow a domain expert to implement a retrieval system based on this methodology.

1. Introduction

Traditionally, access to information stored in online text has been approached using a variety of information retrieval techniques. [SALT83] For the most part, current commercial retrieval systems are implemented by constructing records to represent each document. These records may include keywords or phrases using a free vocabulary (and a thesaurus) or a controlled vocabulary. Records are constructed either by automatic indexing techniques that use word frequency and clustering procedures or by human indexers who choose indices after reading the material. There do exist a few systems that index documents on every non-stoplisted word in the document. Stoplisted words are excluded because of their lack of information content and topic discrimination. Large inverted files are used to quickly locate the non-stoplisted word positions and indexed documents.

Some inherent problems plague the use of these systems because of their design. The interfaces to these systems usually consist of interactive environments in which the user requests documents by posing queries in a retrieval language based on Boolean algebra. Boolean algebra, unfortunately, is not a natural form of expression, so neophytes must spend considerable time and effort to master it.

Another weakness of key-word retrieval systems is the inability of flat records to represent conceptual relationships. For example, the query

damage AND usufructuary AND property AND hurricane

is intended to locate cases where the usufructuary (person who holds limited property rights) is responsible for the damage caused to property by a hurricane. Unfortunately, the query does not establish the semantics of the legal situation at all and thus calls for a blind search for documents that contain all four key words. A key-word system using this query is likely to retrieve irrelevant documents and to miss important ones that specify conceptual variants, such as "tropical storm" for "hurricane." The problem of choosing the actual synonym used for indexing requires the use of a thesaurus. Having a thesaurus available is at best a partial solution since a thesaurus does not represent the root meaning of the concept of "storm".

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

2. The COREL System

COREL is an experimental prototype information retrieval system designed to operate in the legal domain. We have attempted to incorporate some artificial intelligence techniques to improve "accessibility" for users not well acquainted with the complex inner workings of traditional retrieval systems. We believe that the legal domain is a good choice for an attempt of this type since, by nature, it is highly conceptual and contains some narrow areas of concentration.

COREL addresses the problems of traditional retrieval systems primarily by providing a rich array of *knowledge structures* that are used to index articles of the Civil Code of Louisiana. Each structure represents a legal concept. The example below shows how COREL exploits the conceptual nature of the legal domain. The concept of *damage* has associated with it several conceptual attributes that are represented in *slots* and incorporated into a CONCEPT frame. When damage occurs, it is important to know how it occurred, who caused it, and who is responsible for its repair or reparation. These items are included in the DAMAGED-THING frame, set forth below in Figure 1.

DAMAGED-THING
who or what caused damage:
how damage was caused:
who is responsible for repair:

Figure 1: Damaged-Thing concept frame

Unlike key-word systems that require the user to determine which keywords are most likely to retrieve the desired information, COREL guides the user through a set of predetermined *conceptual* frames to locate the sought after information. Because this system has been implemented in the legal domain, a specialized legal vocabulary to describe CONCEPT slots and fillers is necessary and has been added to the system. The user is not required to discover which words have been chosen to describe this information.

* Author's current address: JAYCOR, c/o Naval Research Laboratory, NCARAI - Code 7510, Washington, D.C. 20375

The construction of a complex query with Boolean connectives is not required. Discrimination nets [CHAR80] are used to guide the user in the query process. The use of discrimination nets also provides an automatic and *immediate* narrowing of the feasible areas of retrieval. This is an improvement over Boolean query systems since the user will always have some idea of the direction in which the search is proceeding because the semantics of the user's replies lead the search. There is no need to retrieve a possibly enormous set of documents to verify the effectiveness or ineffectiveness of keyword choices. At each point in the search process, a conceptual frame is activated and displayed to aid the user in the navigation of the search.

3. Things COREL Does Not Do

COREL is not a good general-purpose, multiple domain retrieval system. Its power is achieved because it is not forced to deal with a hodge-podge of documents. Keyword systems are better at dealing with a wide range of topics. It would be a great stride for technology to overcome these problems for the larger systems. The point is that it is worthwhile to design good retrieval systems to cover smaller topic areas.

COREL does not have the ability to reason about the legal concepts contained in its database. It is necessary that the user know what legal concept he or she is pursuing. It is not possible to describe a legal situation to COREL and then have it determine what legal concept is at hand. It acts as a navigational aid for the user in locating relevant pieces of law by following legal concepts. The capabilities mentioned above are being addressed in our larger project called CCLIPS (Civil Code Legal Information Processing System) [DEBE85a, DEBE85b, CROS85] which has as its goals the retrieval of laws relevant to fact situations, legal drafting, and the development of a general representation language for legal concepts. Building a navigational aid for legal concepts is a subgoal of the overall CCLIPS effort.

Finally, the concepts that form the discrimination net must be hand-crafted for each domain. The usefulness of the system depends on how well the database represents the concepts of the domain and is therefore dependent on the system designer. The system builder must understand the domain well to formulate a usable conceptual model.

4. The Evolution of COREL

We began our investigation by classifying the types of articles that appear in the Louisiana Civil Code. Two basic types were chosen to model the conceptual structure of the system. The first type defines terms, and the second type describes legal relationships. Relationships are further subdivided into relationships between people and relationships involving *things* (like property). Within sections about specific relationships, articles that describe the relationship, articles that deal with the rights and duties of the *actors* in the relationship, and if the relationship involves a *thing*, there are articles that deal with the physical and legal attributes of the *thing*.

Because this kind of information appears throughout the Civil Code, it was necessary to design knowledge structures that would support the entailed legal concepts. The means for creating structures for this information have been incorporated into the prototype system. A major influence in choosing these knowledge structures was the organization of the written Civil Code itself. This is not surprising since the document is organized in such a way that conceptually similar relationships are often placed in proximity. If it were not this way, it would be difficult to locate anything.

Once a basic model for a system like COREL was created, the question of how sophisticated the discrimination net should be arose. Decisions had to be made about what level of detail should be used in the nets and about how many articles should be allowed to reside in the same *conceptual* bucket (in the sense of hashing). We found that at a certain level of granularity, the conceptual nature begins to degrade. That is, there exists a point at which the details of the law must be used. Because the goal of CCLIPS is to be able to converse in detail with a user about legal situations, the decision was made to implement the system using little factual information. As a result, there are usually less than five articles indexed under a single bucket.

The Louisiana Civil Code consists of about 3500 articles, each of which on the average is the size of a paragraph. For the COREL prototype, we limited our attention to the section dealing with *things* and to the concept of *usufruct*.

A liberal definition of usufruct is as follows. Usufruct is a relationship between at least two people, one called the *usufructuary* and the other called the *naked owner*. One of them may also be the *grantor* of the usufruct. We assume that the *naked owner* and the grantor are the same person. The grantor grants the right of total use and enjoyment of some *thing* and its *fruits* to the usufructuary. The naked owner retains all other rights to the thing (such as right to alienate *i.e.* the right to convey the title). Fruits are things such as crops, dividends or revenues, and timber. The relationship of usufruct exists for a limited time only.

5. The Structure Of The Prototype System

COREL is written in PEARL [DEER81] and FRANZ LISP [FODE80] and runs on a Vax 11/780. This section will describe the knowledge structures that have been used in the prototype. The following section will describe the package for implementing a COREL system.

The discrimination net used in the system consists of nodes of several types. The top level node is really only a menu that lists all of the TYPE frames available in the system. These frames represent groups of legal relationships that have similar legal purposes. There are five different TYPE frames defined in COREL for describing groups of relationships that have one or more of the following legal purposes:

1. To establish some rights less than ownership of a thing
2. To establish or exchange the rights of ownership to a thing
3. To create and/or describe family relationships
4. To form a cooperative effort to do business
5. To benefit or protect the general public.

The next example (Figure 2) describes type number one. It is the SUBTYPE-LESS-THAN-OWNERSHIP type. This particular type indexes relationships such as usufruct where some rights to a thing are being temporarily transferred to another person or persons. The rights being transferred do not cause the receiving persons to become the owner of the thing, hence the name of this type. The slots appearing in this frame help to further

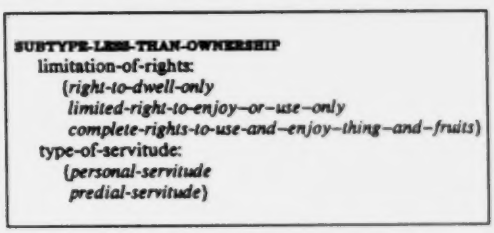


Figure 2: Type classifier structure

discriminate relationships with the goal of establishing rights in a thing less than ownership. The bracketed lists are the allowable slot entries. We mean servitude in the legal sense: an object is subject to a certain use or enjoyment by another person or for the benefit of another person. Predial means pertaining to land or other immovable property.

SYSTEM-DESCRIPTION-RELATIONSHIP frames are "indexed" by the TYPE frames and contain some information about individual relationships. Their purpose is to provide a *context* for the CONCEPT frames that belong underneath it in the net. The next example (Figure 3) is a frame for the *usufruct* relationship. Notice the slots in this type of frame are basic and are more "record-like" in the traditional sense. They are called SYSTEM descriptions because the system makes use of the information in its slots further along in the searching process. From this information, the system is able to provide a list of possible actors that play roles in the other CONCEPT frames.

```
SYSTEM-DESCRIPTION-RELATIONSHIP
   relationship name:
      usufruct
   list of actors:
      grantor
      usufructuary
      naked owner
   type name:
      SUBTYPE-LESS-THAN-OWNERSHIP
```

Figure 3: System Description Relationship frame

Once a relationship type has been identified, the user is ready to move on to the CONCEPT frames. CONCEPT frames are like the DAMAGED-THING frame that was described in a previous section. These frames constitute the bulk of the system. When the user has reached the level of the CONCEPT frames, he or she may either retrieve articles, if any are indexed at this level, or go down to lower level CONCEPT frames. There is no limit to how many levels of concepts that may be contained in the net. However it is our experience that few levels are necessary. Figure 4 shows how the DAMAGED-THING concept frame would appear indexed under the *usufruct* relationship. Under each slot only one option can exist in a CONCEPT frame. All the available options for usufruct have been displayed here for clarity.

```
DAMAGED-THING
   how-damaged:
      (normal-use-wear-or-decay
       abuse
       neglect
       accident-with-someone-at-fault
       no-fault-accident-or-natural-accident)
   responsible-party:
      (usufructuary
       naked owner
       no one)
   who-make-repair:
      (usufructuary
       naked owner
       grantor
       no one)
   rel-name:
      (article number)
```

Figure 4: Damaged Thing concept frame for usufruct

The last object in the system is the DEFINITION structure. The only semantic link that definitions have with relationships is a field that specifies in what context the definition is made. These structures are the simplest type in the system. The only concept they embody is that of making a definition. The general outline of a DEFINITION is given in Figure 5. The lines prefixed with semicolons are LISP comments.

```
DEFINITION
   relationship name:
      ; this describes the context
      ; in which the definition was made
   word:
      ; the word or phrase being defined
   article number:
      ; the article which sets forth definition
```

Figure 5: Definition frame

6. The COREL Definition Language Packages

Some parts of COREL are mutable, whereas others are brittle and unchangeable. The structures for SYSTEM-DESCRIPTION-RELATIONSHIPS and those for DEFINITIONS are hard coded. However, there are three program generation packages in COREL that can be used to expand the coverage. These are LISP macros that can be used to add new knowledge and conceptual structures to the system. They are: Makemenu, Makeconcept, and Makesubtype. Any movement of the COREL domain knowledge can be accomplished by the appropriate coding of the calls to these macros. These are *all* that one needs to remodel or add to the system. This section will describe how one makes calls to the LISP package and what the syntax of these statements looks like.

COREL is a *menu-driven* system. When the user interacts with COREL, menus asking the user to choose options from a list appear on the screen. These menus constitute the discrimination net. The menus ask questions that fill in slots of the previously described structures. Once an entire frame has been filled in (possibly with "I don't know" answers) a new set of more specific objects must be identified, if possible. At this point the system moves on to another menu and asks relevant questions about these new objects in an attempt to narrow down the number of choices. Menus are defined by the MAKEMENU macro. This statement allows the designer to list a menu description or question and the allowable responses. Associated with each allowable response is another menu call, a LISP function call, or a LISP PROG body to be executed if that choice is made. Figure 6 gives the BNF description of the syntax for the MAKEMENU statement. Figure 7 is an example of a MAKEMENU statement used in the prototype system.

```
<makemenu>      ::=  (makemenu
                        <menu-name>
                        <explanation>
                        <description>)

<menu-name>     ::=  <atom>

<explanation>   ::=  <string>
                ::=  <string> <explanation>

<description>   ::=  (<explanation> <action>)
                ::=  (<explanation> <action>) <description>

<action>        ::=  <other-menu-name>
                ::=  <function-name>
                ::=  <lisp-prog-body>
```

Figure 6: Syntax of the MAKEMENU statement

```
(makemenu   menu6
   ("    DETERMINATION OF CONCEPTUAL GROUP"
    "                                      "
    "The following are the main conceptual categories"
    "available for relationships with the primary pur-"
    "pose of establishing rights less than ownership")

   ("Information about any alteration of the physical state"
    "of the thing"
      menu7)

   ("Information about the legal relationship"
      menu8)

   ("Information about the rights and duties of the actors"
      menu9))
```

Figure 7: Example of a MAKEMENU statement

The MAKESUBTYPE package describes TYPE frames. The syntax provides the user with the ability to specify a slot name, an explanation string for the slot, and a list of options. Some slots, depending on the answers given, may be filled with something other than the listed options. An example of this can be seen under the *type-of-servitude* slot. It is also possible to fill in slots in the structure based on values of previously filled slots. This is illustrated in the example of the MAKESUBTYPE package shown in Figure 9 (the syntax is given in Figure 8).

```
(makesubtype  LESS-THAN-OWNERSHIP
     (limitation-of-rights
          "The right(s) to be established in the thing are:"
               right-to-dwell-only
               limited-right-to-enjoy-or-use-only
               complete-rights-of-use-and-enjoyment-of-thing
               -and-fruits)

     (type-of-servitude
          "The right(s) will be established in favor of a:"
               natural-or-legal-person
               immovable-thing
                    (if natural-or-legal-person
                     then
                          (fill type-of-servitude
                               personal-servitude))
                    (if immovable-thing
                     then
                          (fill type-of-servitude
                               predial-servitude))
     menu6))
```

Figure 9: Example of a MAKESUBTYPE statement from COREL

The MAKECONCEPT statement is similar to the MAKESUBTYPE statement. However, since the programs generated by the two different macro packages differ, these must remain two separate statements. The syntax of the statement (Figure 10) and an example from the COREL implementation follow (Figure 11).

<makesubtype>	::=	(**makesubtype** <type-name> <slot-descriptions> <menu-connection>)
<slot-descriptions>	::=	<slot-body>
	::=	<slot-body> <slot-descriptions>
<slot-body>	::=	(<slot-name> <question-string> <value-list> <conditional>)
<type-name>	::=	<atom>
<slot-name>	::=	<atom>
<question-string>	::=	<string>
	::=	<string> <question-string>
<value-list>	::=	<pearl-symbol>
	::=	<pearl-symbol> <value-list>
<conditional>	::=	<cond-body>
	::=	**null**
	::=	<cond-body> <conditional>
<cond-body>	::=	(**if** <cond> **then** <then-body>)
<cond>	::=	<slot-value>
	::=	**true**
<then-body>	::=	(**fill** <slot-name> <pearl-symbol>)
	::=	<lisp-function>
<slot-value>	::=	<pearl-symbol>
	::=	**true**
<menu-connection>	::=	<menu-name>
	::=	<lisp-statement>

Figure 8: Syntax of the MAKESUBTYPE statement

<makeconcept>	::=	(**makeconcept** <concept-name> <slot-descriptions>)
<slot-description>	::=	<slot-body>
	::=	<slot-body> <slot-description>
<slot-body>	::=	(<slot-name> <question-string> <value-list> <conditional>)
<concept-name>	::=	<atom>
<slot-name>	::=	<atom>
<question-string>	::=	<string>
	::=	<string> <question-string>
<value-list>	::=	<pearl-symbol> <PV2>
	::=	<automatic-actor-list> <PV2>
<PV2>	::=	<pearl-symbol> <PV2>
	::=	**null**
<conditional>	::=	<cond-body>
	::=	**null**
	::=	<cond-body> <conditional>
<cond-body>	::=	(**if** <cond> **then** <then-body>)
<cond>	::=	<slot-value>
	::=	**true**
<then-body>	::=	(**fill** <slot-name> pearl-symbol)
	::=	<lisp-function>
<slot-value>	::=	<pearl-symbol>

Figure 10: Syntax of the MAKECONCEPT statement

```
(makeconcept DAMAGED-THING
  (how-damaged
    "The damage occurred from:"
      normal-use-wear-or-decay
      abuse
      neglect
      accident-with-someone-at-fault
      no-fault-accident-or-natural-accident
        (if normal-use-wear-or-decay
        then
          (fill responsible-party
            no-one-at-fault))
        (if no-fault-accident-or-natural-accident
        then
          (fill responsible-party
            no-one-at-fault)))
  (responsible-party
    "Who is responsible for the damage?"
      <automatic-actor-list>
      other-person)
  (who-make-repair
    "The thing must be replaced, repaired, or paid for by:"
      <automatic-actor-list>
      other-person
      no-one))
```

Figure 11: Example of a MAKECONCEPT statement

The *if* statements in the *how-damaged* slot illustrate how the system automatically fills slots based on other given information when it is possible to do so. After the slot is filled by one of these statements, the user will not be queried for another response for that slot. The <automatic-actor-list> is expanded to be a list of actors mentioned in a previous part of the dialog.

7. Future Directions And Interesting Problems

The current COREL system is small. The next step in this research project will be to expand the number of CONCEPTS so that more articles of the Civil Code can be indexed. This step should be performed under the supervision of a domain expert.

Once the system has been enlarged to contain a reasonable number of articles, we will compare and contrast the system with key-word systems. The techniques incorporated into the system intuitively seem to be better than key-word retrieval techniques, but empirical data should be collected.

From the perspective of artificial intelligence, we want to explore the possibility of automatically indexing articles using natural language techniques and concept frame directed understanding. This endeavour is a major undertaking and would constitute a large research project by itself.

Acknowledgements

We gratefully acknowledge the support of the Louisiana State Law Institute (principal sponsor), the Center of Civil Law Studies of the LSU Law Center, and the Southern University Law School in conducting this research. We also thank Professor Donald Kraft for reviewing an earlier draft of this paper.

References

CHAR80.
Charniak, Eugene, Riesbeck, Christopher K., and McDermott, Drew V., *Artificial Intelligence Programming*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1980.

CROS85.
Cross, George R. and deBessonet, Cary G., "Representation of Legal Knowledge for Conceptual Retrieval," *Information Processing and Management*, vol. 21, no. 1, pp. 35-44, 1985.

DEBE85a.
deBessonet, Cary G. and Cross, George R., "An Artificial Intelligence Application in Law: CCLIPS, A Computer Program that Processes Legal Information," Technical Report TR-85-001, Louisiana State University, Computer Science Department, Baton Rouge, LA, 1985.

DEBE85b.
deBessonet, Cary G. and Cross, George R., "Representation of Some Aspects of Legal Causality," in *Computing Power and Legal Reasoning*, ed. Charles Walter, pp. 205-214, West Publishing Company, St. Paul, 1985.

DEER81.
Deering, Michael, Faletti, Joseph, and Wilenksy, Robert, "PEARL: An Efficient Language for Artificial Intelligence Programming," *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pp. 930-932, Vancouver, British Columbia, August, 1981.

FODE80.
Foderaro, J. K., *The Franz LISP Manual*, Regents of the University of California, Berkeley, CA, 1980.

SALT83.
Salton, Gerard and McGill, Michael J., *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, 1983.

HIERARCHIC DOCUMENT CLUSTERING USING WARD'S METHOD

A. EL-HAMDOUCHI and
P. WILLETT
Sheffield University,
Western Bank,
Sheffield, S10 2TN, UK

ABSTRACT.

In this paper, we discuss the application of a recent hierarchic clustering algorithm to the automatic classification of files of documents. Whereas most hierarchic clustering algorithms involve the generation and updating of an inter-object dissimilarity matrix, this new algorithm is based upon a series of nearest neighbour searches. Such an approach is appropriate to several clustering methods, including Ward's method which has been shown to perform well in experimental studies of hierarchic document clustering. A description is given of heuristics which can increase the efficiency of the new algorithm when it is used to cluster three document collections by Ward's method.

1. INTRODUCTION

The subject of document retrieval involves the study of computerised methods for the storage and retrieval of textual items such as memoranda, technical reports and references to journal articles (VANR79, SALT83). Document retrieval systems have for long been the preserve of the librarian or information worker, but the spread of office automation facilities is leading to increased interest in the use of such systems in a wide range of environments.

Documents are generally represented for retrieval purposes by a series of keywords that have been assigned using either manual or automatic indexing, and similar lists of keywords are used to represent the queries which are presented to the system. The sets of document and query keywords form the basis for a matching operation which identifies those documents that are most similar to the query, and thus those that are most likely to be relevant to the person who submitted the query. A very wide range of methods has been suggested by which a machine-readable document file may be structured to facilitate the query-document matching operations, and one such application involves the use of document clustering techniques.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.
© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

Document clustering involves the application of a clustering method to the lists of keywords representing each of the documents in a collection, resulting in the identification of clusters of documents that have many terms in common and that might accordingly be expected to be about the same, or related, subjects. A query may then be matched against the set of clusters, rather than against each of the documents. Not only may this increase the efficiency of a search, since fewer matching operations are required, but it may also increase the effectiveness of a search, in terms of the amount of relevant material that is retrieved, since the inter-document term similarities have been taken into consideration in the structuring of the file.

Early work on document clustering focussed upon the use of non-hierarchic partitioning algorithms which permitted a rapid clustering of a file, but which were found to give classifications that were less effective for retrieval purposes than conventional, non-clustered search strategies (SALT71). Accordingly, attention has focussed upon the use of hierarchic clustering methods, specifically hierarchic agglomerative methods, which result in binary tree-like classifications in which small clusters of strongly related documents are nested within larger and larger clusters of less strongly related documents.

The first detailed studies of the use of hierarchic agglomerative methods for document clustering was by Jardine and van Rijsbergen (JARD71b) and van Rijsbergen (VANR74) who used the single linkage method to cluster several sets of documents and queries: further experiments with the single linkage method have been described by van Rijsbergen and Croft (VANR75) and Croft (CROF77, CROF80). More recently, Griffiths et al. (GRIF84, GRIF86) have described comparative experiments which involved not only single linkage but also the complete linkage, group average and Ward, or minimum variance, methods. Their results suggest that the single linkage method is the least appropriate of the methods studied, with the most useful results generally being obtained from the use of Ward's method. This method was found to

result in balanced hierarchies that gave high levels of retrieval effectiveness in a range of cluster searches based upon the bottom level clusters in the hierarchy. However, the experiments were restricted to collections containing 800 documents or less, and while such a file is large in comparison with the data sets that are encountered in most applications of hierarchic cluster analysis, it is trivially small in comparison with the file sizes encountered in document retrieval systems which may contains tens or hundreds of thousands of documents. Thus although the experimental results to date have suggested that document clustering may indeed have substantial merits for retrieval purposes, there is considerable scepticism as to the use of document clustering on files of realistic size. A notable exception is the work of Croft [CROF77] who has reported document clustering experiments using part of the UKCIS document test collection: however, he used only the single linkage method, and till recently there have been no algorithms available which would permit other hierarchic clustering methods, such as Ward's method, to be applied to data sets of comparable size. However, several new and efficient algorithms for hierarchic agglomerative clustering have been reported recently which are based upon series of repeated nearest neighbour searches [DAYE84, MURT84a, MURT84b]. In this paper, we study the application of these algorithmic developments to automatic document clustering using Ward's method; the description is at a sufficient level of detail to allow other workers to implement this clustering method without undue difficulty should they so desire. We have used the resulting program in an extended comparative study of hierarchic clustering methods; the results of this study will be reported in a future communication.

2. HIERARCHIC AGGLOMERATIVE CLUSTERING

2.1 Clustering algorithms

The traditional algorithm for obtaining a hierarchic agglomerative classification of a collection of N points is as shown below: we shall refer to this as algorithm A. The reader should note that the term 'point' refers initially to individual documents but encompasses clusters of documents as the agglomeration proceeds.

```
Calculate all N(N-1)/2 distinct inter-point
  dissimilarities and store them in an
  inter-point dissimilarity matrix ;
WHILE more than one point remains DO
  BEGIN
    Search the matrix to identify the least
      dissimilar remaining pair of points ;
    Fuse these two points to form a new point ;
    Update the matrix by calculating the
      dissimilarities between this new point
      and each of the other points
  END.
```

It will be seen that algorithm A involves the generation of a matrix which initially contains the dissimilarities between all pairs of documents in the collection. The matrix is repeatedly scanned to identify the least dissimilar pair of points, and these are then fused together and the dissimilarities calculated that involve this new point. Different clustering methods may be accomodated within this general framework by varying the precise definition that is used in the calculation of the dissimilarities [LANC67]. Since there are $N(N-1)/2$ entries in the dissimilarity matrix, this algorithm has a storage requirement of $O(N^2)$. The main loop of the algorithm is performed N-1 times, once for each cluster formed during the generation of the hierarchy, and each such iteration may require the scanning of all of the $O(N^2)$ entries in the matrix to identify the least dissimilar pair of points; thus the time requirement of the algorithm is $O(N^3)$. More efficient algorithms are known for certain specific methods. Thus Sibson, [SIBS73] has described an $O(N)$ space and $O(N^2)$ time algorithm for the single linkage method and Defays [DEFA77] has reported a modification of the Sibson algorithm which may be used to generate a non-unique complete linkage hierarchy: in both cases, the dissimilarities in the matrix are calculated one row at a time and are used to update the hierarchy recursively. However there are no comparable algorithms for Ward's method.

The new algorithms for hierarchic agglomerative classification are well reviewed by Murtagh [MURT83, MURT84a] who emphasises the central importance of nearest neighbour (NN) searching, where the nearest neighbour of some point I, NN(I), is defined to be that point which is least dissimilar to I. Then an NN-chain is defined to be the sequence of points I, J=NN(I), K=NN(J)...Y=NN(X), X=NN(Y). X and Y are said to be reciprocal nearest neighbours, or RNNs. We note in passing that the use of NN-chains for document retrieval has been described by Goffman [GOFF69]. The particular algorithm studied here, algorithm B, is as follows:

```
Select some arbitrary point I ;
WHILE there is still more than one point DO
  BEGIN
    REPEAT
      J=NN(I), K=NN(J), L=NN(K) etc.
    UNTIL Y=NN(X) and X=NN(Y) ;
    Fuse X and Y together and update the data
      matrix by replacing them with the
      centroid of the new point ;
    I := either the point prior to X in the
      NN-chain or, if no such point exists,
      an arbitrary point
  END.
```

Murtagh [MURT83] describes this as a single cluster algorithm for geometric clustering methods. It is a single cluster algorithm since only one NN-chain, and hence only one cluster, is grown at a time: this is in contrast with multiple cluster algorithms where several NN-chains may be grown simultaneously. Geometric clustering methods are those where the new point that is formed by the fusion of two previous points can be represented by a centroid, or point centre: this is not the case with the alternative linkage methods such as single or complete linkage. Examples of clustering methods that can be implemented by algorithm B include the centroid, median and Ward methods: they may be obtained merely by altering the dissimilarity measure used to generate the NN-chains, and the definition of the point centre used to characterise each new point as it is formed. If the current point in the NN-chain, i.e. the point whose nearest neighbour is being sought, is called the query point, Q, a Ward classification may be obtained by, firstly,

setting the dissimilarity, D(Q,I), between Q and the I'th point in the collection (I <> Q) equal to

$$\frac{SIZE(Q)*SIZE(I)*DIST(Q,I)}{SIZE(Q)+SIZE(I)}$$

where SIZE(Q) and SIZE(I) are the numbers of documents in the points Q and I, and where DIST(Q,I) is the Euclidean distance between them, and, secondly, defining the point centre representing the point arising from the fusion of two points Q and I by the expression

$$\frac{SIZE(Q)*VEC(Q)+SIZE(I)*VEC(I)}{SIZE(Q)+SIZE(I)}$$

where VEC(Q) and VEC(I) are the index term vectors characterising the points Q and I.

2.2 Nearest neighbour searching

It will be seen that the repeated scanning and updating of the dissimilarity matrix in the simple algorithm A is replaced in algorithm B by a sequence of N-1 RNN searches. Thus the overall running time will be crucially dependent upon the efficiency of the search procedure that is used to identify the nearest neighbours at each stage of the clustering.

The obvious, brute-force algorithm for nearest neighbour searching is to compute the dissimilarity between the current point in the NN-chain, Q, and each of the other points in the file, a procedure that is much too slow for all but the smallest data sets. To overcome this problem, a range of procedures has been suggested that avoid the calculation of most of the dissimilarities while still calculating them for those few points which are, in fact, similar to the query: examples include the projection of the multi-dimensional points onto a space of lower dimensionality, where most of the dissimilarity calculations are performed, and the grouping of points into clusters so that several points may be searched, or eliminated from the search, simultaneously [Murtagh, 1984b].

Unfortunately, most of the nearest neighbour algorithms in the literature are not directly applicable to searching and clustering files of documents since they generally assume that H, the dimensionality of the space that contains the records, is very low, typically 2 or 3, so that multiplicative terms involving H in the equation describing the complexity of the search algorithm may be neglected. In text retrieval systems, conversely, many thousands or tens of thousands of keywords may be used to characterise the documents in the file, and such algorithms are thus quite impracticable. An example of this is the O(logN) nearest neighbour search procedure of Friedman et al. [FRIE77] which Weiss [WEIS81] suggests involves a constant of proportionality of about 1.6^H. Again, the constant expected time search algorithm of Bentley et al. [BENT80] involves the inspection of all of the 3^H-1 cells adjacent to a given cell in the space, an undertaking that is quite infeasible if the dimensionality is at all large, and also the assumption that the data records have been chosen independently from a uniform point distribution throughout the space. As a further example of such problems, Kittler [KITT78] describes a heuristic for nearest neighbour searching using the city block distance, the efficiency of which is proportional to $H^{-1/2}$.

For these reasons, considerable work has gone into the design of nearest neighbour algorithms that could be used with the high dimensionality data which characterise textual data bases. The majority of the algorithms which have been developed involve the use of the inverted file structure [NORE77, SMEA81, MURT82, PERR83], and we have used one such algorithm as the basis for a program for Ward's clustering method. The exact manner in which this is achieved is described in the next section of the paper.

3. THE PROGRAM

3.1 Introduction

The strategy that is chosen for NN-searching will depend critically upon the inter-document dissimilarity measure used. In the case of Ward's clustering method, the dissimilarity D(Q,I) between the query point Q and some arbitrary point I is as given in Section 2.1 above. For I to be the NN of Q, two situations may be identified, depending upon whether Q and I do or do not have terms in common.

It is normally assumed in document clustering that documents which have no terms in common should not be considered as being related in any way. However, such a restriction may mean that the true NN for some Q may not be identified, resulting in an approximate and inexact hierarchy; as we show below in Section 4.1, NNs which do not have terms in common with the current Q occur quite frequently in at least one of the data sets which we have studied. Moreover, if such NNs are ignored, the situation can, and does, arise that it is impossible to create a full hierarchy; instead, a series of discrete sub-hierarchies is created which cannot be linked into a single classification. In the practical context of a clustered document file, this need not be too serious a problem if bottom-up cluster searches are employed since it is then unlikely that the upper reaches of the hierarchy will need to be inspected [CROF80]: this will not be the case if top-down algorithms are used to search the classification [VANR74]. Accordingly, we shall assume for the moment that an exact hierarchy is to be generated in which NN(Q) is allowed to have no terms in common with Q. We shall refer to the set of points having terms in common with Q as COMM, and the set of points having no terms in common with Q as NCOMM. Then the identification of NN(Q) reduces to that of identifying NN(Q)C, the NN for COMM, and NN(Q)NC, the NN for NCOMM, and comparing the two dissimilarities to identify the true nearest neighbour. The identification of NN(Q)C and NN(Q)NC are described in Sections 3.3 and 3.4 below. However, before doing so, it is convenient to list briefly the main data structures that are required by our program.

3.2 Main data structures

Each of the N documents in the collection which is to be clustered is characterised by a list of integers, corresponding to the indexing terms which have been assigned to the documents. The set of N such lists comprises the basic data matrix DOCFILE. The document collections considered in this paper all involve binary, i.e. present or

absent, indexing without the use of any within-document weighting scheme. If such a scheme were to be used, DOCFILE would need to contain the weight of each term in each document, in addition to its presence.

The inverted file, INVFILE, consists of a set of M lists, one for each of the terms in the vocabulary that has been used to index the documents. The I'th such list (1<=I<=M) contains the identifiers for those points which contain the I'th term, these identifiers being accompanied by a term coefficient which denotes the weight of the term in that point. Initially, these coefficients are all equal to unity and the point identifiers are in the range 1 to N, corresponding to individual documents in the collection. As the agglomeration proceeds, however, new points are created, with identifiers in the range N+1 to 2N-1, and this fact is reflected by updates to INVFILE (as described in Section 3.6) which change both the point identifiers and the accompanying coefficients.

Several other fixed-size arrays are required by the program. SIZE is an integer array in which the I'th element contains the number of documents in the I'th point. If the initial data set contains N documents, a total of N-1 new points will be created, and thus SIZE contains a total of 2N-1 elements. SUMSQ is a real array in which the I'th element contains the sum of the squares of the coefficients for the terms in the centroid representing the I'th point: this again contains 2N-1 elements. It should be noted that the centroids are not stored explicitly owing to the excessive amounts of internal storage that would be required; instead centroids are calculated as and when required, i.e. when a point becomes the current query point whose NN is required and when an RNN-pair has been identified. The calculation of a centroid is achieved by use of the array RNN. This is an (N-1)x3 array in which the I'th trio of elements contains the identifiers for the points involved in the creation of the I'th NN-pair, together with the dissimilarity at which this pair was created: this latter information is not actually required by the clustering program, but forms an important part of the routines which are used to carry out searches of the cluster hierarchy once it has been generated. To identify the individual documents comprising the I'th point, for I>N, RNN(I-N) is accessed to obtain a pair of identifiers. If either of these is greater than N, P say, then RNN(P-N) is accessed to identify the points that fused to form P and so on until identifiers <= N, i.e. individual documents, are obtained. Once the set of documents comprising the I'th point has been identified in this manner, DOCFILE may be accessed to obtain the corresponding term vectors and the centroid then calculated.

It is assumed that all of these arrays and lists can be fitted into the main store, real or virtual, of the machine which is being used for the clustering. If this is not the case, the program will entail excessive run-times owing to the very large number of disc accesses required.

3.3 Identification of NN(Q)C

The inverted file nearest neighbour algorithms mentioned in Section 2.2 all involve consideration of the points in COMM, and thus result in the identification of NN(Q)C. A critical review of these algorithms [PERR83] suggested that the most efficient algorithm was that due to Noreault et al. [NORE77] and we have used this in our clustering program for the identification of NN(Q)C. The algorithm involves the processing of the point identifiers and term coefficients in the inverted file lists corresponding to the terms in the query point, which we shall refer to as query lists. The processing of the query lists results in a new list containing the identifiers of all of those points that have at least one term in common with the query, together with the dot product between the term vectors representing Q and each other point. The dot products may be calculated as follows. When a point identifier, I say, is encountered for the first time in a query list, a counter, DOTPROD(I), is allocated to that point and set equal to the product of the term coefficients for I and for Q; DOTPROD(I) is incremented by further such products each time that I is encountered in subsequent query lists. When all of the lists have been processed in this way, each counter will contain the sum of the term coefficient products between the query and one of the points in the collection that is being searched. These sums may then be used for the calculation of the distance DIST(Q,I) and hence the dissimilarity D(Q,I): the D(Q,I) values may then be used to identify the nearest neighbour of Q, NN(Q)C. This procedure may be summarised by the following pseudo-code: TC(Q) and TC(I) are the term coefficients for Q and I in the current query list, and MINDISSIM is the smallest dissimilarity value calculated at any stage of the NN-search

```
FOR each query list DO
  BEGIN
    FOR each point, I, in this query list DO
      IF identified in a previous list THEN
        DOTPROD(I) := DOTPROD(I) + TC(Q)*TC(I)
      ELSE DOTPROD(I) := TC(Q)*TC(I)
  END ;
NN(Q)C := 0; MINDISSIM := MAXINT ;
FOR each point, I, in COMM DO
  BEGIN
    DIST(Q,I) :=
      SQRT(SUMSQ(Q)+SUMSQ(I)-2*DOTPROD(I)) ;
    D(Q,I) := SIZE(Q)*SIZE(I)*DIST(Q,I)
    D(Q,I) := D(Q,I)/(SIZE(Q)+SIZE(I)) ;
    IF D(Q,I) < MINDISSIM THEN
      BEGIN
        NN(Q)C := I ;
        MINDISSIM := D(Q,I)
      END
  END .
```

3.4 Identification of NN(Q)NC

When Q and I share no common terms, DOTPROD(Q,I) is zero and thus the expression for D(Q,I) reduces to

$$\frac{SIZE(Q)*SIZE(I)*SQRT(SUMSQ(Q)+SUMSQ(I))}{SIZE(Q)+SIZE(I)}$$

Accordingly, D(Q,I) will be monotonically increasing with both SIZE(I) and SUMSQ(I), and this fact can be used to eliminate many documents from consideration as NN(Q)NC. The search may be

restricted as follows

1. rank all points in NCOMM in order of increasing SUMSQ(I)
2. identify the smallest value of SIZE(I) in NCOMM
3. compare Q with the points in NCOMM in order of increasing SUMSQ(I) until a point I is reached for which SIZE(I) equals the smallest value noted in 2

NN(Q)NC is then the point corresponding to the smallest dissimilarity identified upto I.

We note that if NN(Q)C has already been identified, a simple lower bound calculation may be undertaken to determine whether we actually need to identify NN(Q)NC. Specifically, the smallest SUMSQ(I) and smallest SIZE(I) values in NCOMM are used to calculate D(Q,I): only if this lower bound dissimilarity is less than the dissimilarity corresponding to the previously identified NN(Q)C do we need to consider the points in NCOMM.

3.5 Identification of RNN-pairs

A series of NN-searches is carried out as described above to create the NN-chain, and this chain is grown until an RNN-pair is found, i.e. a pair of points X and Y in the chain such that Y=NN(X) and X=NN(Y). There is generally no problem associated with the identification of such a pair. However, consider a sequence such as I=NN(Q), J=NN(I)...Y=NN(X), Z=NN(Y), X=NN(Z) where the NN-chain terminates in a loop. In such cases, means must be provided to specify which point should be taken with X in the RNN-pair. The procedure adopted here is to take as NN(X) the point corresponding to min(SIZE(Y), SIZE(Z)), or, if of equal SIZE, the first point encountered. The latter course introduces a small degree of order dependence into the classification that is produced: it should be emphasised that this arises not from the RNN-based algorithm, or from our implementation of it, but from the fact that nearly all hierarchic agglomerative clustering methods involve some degree of arbitrary choice when equal dissimilarities are identified. Indeed, this undesirable characteristic was one of the reasons why Jardine and Sibson [JARD71a] recommended the use of the single linkage method which is unaffected by equal dissimilarity values.

3.6 Updating operations

Once an RNN-pair has been identified, a series of updating operations is carried out to reflect the fact that a new point has been created as the result of the fusion of two previous points.

When two points, X and Y say, fuse together to form the I'th RNN-pair, the INVFILE lists corresponding to the terms in X and Y are scanned to note all occurrences of the identifiers X and Y. Such occurrences are replaced by N+I, the identifier corresponding to the new point which has been formed: if both X and Y occur in some INVFILE list, only a single occurrence of N+I is stored. INVFILE thus steadily contracts as points are fused together while DOCFILE remains unchanged throughout the entire clustering operation. The term coefficients corresponding to each occurrence of the new point are obtained by identifying the individual documents within the new point: this is done via the array RNN as described in Section 3.2. The term lists in DOCFILE corresponding to these documents can then be used to obtain the centroid of the new point. A knowledge of the individual documents within the new point allows the calculation of SIZE(N+I), the number of documents within it: then if some term occurs in J of them, the coefficient is given by J/SIZE(N+I). Once all of the coefficients in the centroid of the new point have been calculated in this manner and added into INVFILE, the sum of the squares of the coefficients is placed in SUMSQ(N+I). The centroid itself may then be deleted since, as we have noted earlier, it is not required any further, and its storage would entail large overheads. Finally, the I'th element of RNN is filled by storing the identifiers X and Y, together with the dissimilarity level at which they fused.

3.7 Commencing a new NN-chain

Once an RNN-pair has been identified, and the appropriate modifications made to INVFILE, SIZE etc., a decision must be made as to how the RNN-search should be continued. In those cases where the NN-chain contains only two points, i.e. those that fused together to form the RNN-pair, then any arbitrary point can be selected to initiate the new NN-chain. More usually, however, there will be points remaining in the NN-chain after the removal of the two fused points, and we now describe a simple heuristic which can be used to increase the efficiency with which the NN-chain is grown after an RNN-pair has been identified.

Let I, J, K...W, X, Y be an NN-chain in which X and Y form the I'th RNN-pair. Then once they have been identified and fused together, the final point in the chain is W and to commence a new chain it is necessary to identify NN(W). Let us assume that when creating the NN-chain, we identify not just the NNs, but also the point that is second least dissimilar to each point: call this second nearest neighbour NN2. Then it is easy to see that NN(W) can be one of only two points, these being either the new point, N+I, that has been formed by the fusion of X and Y, or NN2(W). Thus if NN2(W) has been stored, the new point in the NN-chain can be identified simply by calculating D(W,N+I) and then comparing this dissimilarity with that corresponding to NN2(W): thus only a single NN calculation needs to be carried out to re-initiate the NN-chain.

More generally, assume that we store not just NN2(W) but also NN3(W), NN4(W)..NNK(W). Then if W subsequently becomes the current point in the NN-chain, NN(W) will be either the highest point in this ranked list of nearest neighbours which has not been clustered since the list was created, or the least dissimilar of the new points which have been identified since the list was created. Thus when some point W is identified for the first time as an NN, we calculate not just NN(W) but also NN2(W), NN3(W) etc. Such ranked lists are constantly updated by deleting any points which have been involved in RNN-pairs, and which have thus been removed from consideration in their original form; in addition, a note is made of those new points which have been created since W

was first processed.

The larger the number of NNs stored for each point, i.e. the larger the value of K, the greater the storage overheads and updating costs; however, this is counter-balanced by the reduction in NN-searching costs that might be expected with a large value for K. The trade-off between these two conflicting factors is investigated further in the next section.

4. EXPERIMENTAL STUDIES

4.1 Results

The program described in Section 3 has been implemented in FORTRAN and run on a Prime 9950 minicomputer. To date, three document test collections have been clustered using this program, these being the Keen, Cranfield and Evans collections whose characteristics are summarised below in Table I.

Each NN-chain terminates in the identification of an RNN-pair, and algorithm B will thus be efficient in operation only if a small number of NN-searches is required for each such pair. In fact, it is easy to show [MURT84a] that the generation of a Ward hierarchy requires a total of 3N-3 NN-searches for a file of N documents, i.e. an average of just under three NN-searches for each RNN pair. That this is so can be seen by noting that the NN-chain representing the complete hierarchy contains the N original points and the N-1 points formed during the generation of the hierarchy, this corresponding to a total of 2N-1 points, and thus 2N-2 NN-searches linking them together. In addition, each of the N-1 RNN-pairs requires an additional NN-search to establish that an RNN-pair has, in fact, been identified. Thus a total of 3N-3 NN-searches is required for the identification of all N-1 RNN-pairs.

The distribution of NN-chain lengths for the identification of the N-1 RNN-pairs for each collection are listed in Table II. The distribution is very similar across the three collections: thus 13.4% (Keen), 12.2% (Cranfield) and 9.7% (Evans) of the RNN-pairs are identified by the minimum number of two NN-searches, while the corresponding figures for three NN-searches is 19.1%, 18.6% and 17.4% respectively. It should be remembered that we are averaging here across widely disparate types of NN-search. At the start of a classification, each point corresponds to an individual document, and a single NN-search might involve consideration of the majority of the other N-1 documents. As the agglomeration proceeds, however, the number of points diminishes so that the final NN-searches involve comparing the query point with only a very few other points. However, as the points grow in size, there will be more and more index terms associated with each point, this leading to an increase in the computational requirements of NN-searching. Thus although there are only a few NN-searches late on in the classification, each of them will be quite time-consuming to effect.

Section 3.7 presented an heuristic which may be used to re-initialise the NN-chain for those cases where one or more points remain in the chain after the detection of an RNN-pair. In such cases, the next NN-search is achieved at a very low computational cost, viz. a single dissimilarity calculation, which is negligible in comparison with the cost of a normal NN-search. There are, at most, N-2 such cases and thus the heuristic might be expected to lead to a reduction of at most one third in the computational requirements of the algorithm (since it normally requires 3N-3 NN-searches as noted above). It was noted in Section 3.7 that a progressive variation in K, the number of NNs stored for each point, might be expected to lead to a well-marked trade-off between reduced NN-searching costs and increased updating costs. Table III gives the running times in seconds of CPU time for the clustering of the three collections as K is varied from 1 upto 10. For the Keen and Cranfield data, the fastest runs are obtained with K=6 while K=4 gives the best results for the Evans collection: these values are starred in the Table. The use of the optimal K value leads to reductions of 25.5% (Keen), 36.7% (Cranfield) and 28.1% (Evans) of the K=1 run times. The value for Cranfield is slightly greater than the theoretical upperbound of 33.3% since the latter figure assumes that all of the NN-searches are of equal cost and, as noted previously, this is not the case.

The experiments with K also considered ignoring the identification of NN(Q)NC during the growth of the NN-chains, i.e. restricting the NN-search to the points in COMM. The results of these experiments suggest that slight reductions in computational costs can be achieved. Thus, for the Keen collection, the running times for K=1, 5 and 10 are 336, 234 and 243 seconds respectively, slightly less than the corresponding figures in Table III. However, the use of this constraint may result in some cases in the true nearest neighbour not being identified for some point: for the collections studied here, 27 (Keen), 13 (Cranfield) and 549 (Evans) of the RNN-pairs did not have any terms in common.

The description of the updating operations in Section 3.6 emphasised the fact that no attempt is made to store the point centroids of the RNN-pairs owing to the storage overheads this would incur. For the collections studied here, there are 7838 (Keen), 40243 (Cranfield) and 16768 (Evans) postings in DOCFILE: an additional 39316 (Keen), 152960 (Cranfield) and 176954 (Evans) postings would be required if the cluster centroids were to be stored explicitly during clustering. If these centroids were stored, of course, DOCFILE would need to contain not only these additional postings but also the corresponding term coefficient values, thus making the overheads still less acceptable.

For comparison with the results obtained here, we have also clustered the three collections using the well-known SLINK algorithm for the single linkage method [SIBS73]. For the Keen, Cranfield and Evans data sets, this algorithm requires 40, 165 and 331 seconds of CPU time respectively, and comparable results are given by the similar CLINK complete linkage algorithm [DEFA77]. The best results from Table III for our algorithm for these three collections are 278, 1356 and 1076 seconds of CPU time. Thus the Ward's method implementation

described here is substantially slower than the other two procedures, the difference being smallest in the case of the Evans collection, where the low indexing exhaustivity permits very rapid NN(Q)C searches to be carried out.

4.2 Increasing the efficiency of clustering

The results of cluster searching experiments suggest that small clusters, containing only a few, closely related documents, are the most effective for retrieval purposes [CROF80, GRIF86]. Moreover, if the effective bottom-up search strategy is used [VANR75], in which the search starts at the small clusters or individual documents at the base of the hierarchy and moves upwards towards the root, it is most unlikely that the larger clusters high up in the tree will need to be matched against a query. There is thus little rationale for devoting a large amount of effort to the identification of the large clusters in the hierarchy. Such clusters, however, require large amounts of computation, this arising from the fact that the efficiency of the NN-searching algorithm involved in the identification of NN(Q)C, as described in Section 3.3, is crucially dependent upon cluster size. A large cluster, i.e. point, containing many documents will have many terms associated with it, and will thus require the processing of many query lists when that point becomes a link in the NN-chain.

These observations suggest that a Ward classification in which the upper reaches of the hierarchy are characterised only approximately, or not at all, might well give levels of retrieval effectiveness comparable to those obtainable from a full hierarchy in which all of the clusters were specified exactly; however, such a classification could be generated at a very much lower computational cost. Two means suggest themselves by which such an approximate hierarchy might be obtained. The first is to adopt the simple expedient of specifying some maximal cluster size or maximal dissimilarity, and then not allowing points to be involved in any further agglomerations that exceeded the chosen threshold. The second strategy is rather less drastic in its handling of the larger clusters and involves the use of a thresholding strategy whenever a point centroid is created, i.e. after the identification of an RNN-pair or prior to an NN-search. Specifically, a term would be considered for inclusion in the centroid if, and only if, it occurred in more than some pre-defined percentage of the documents in the point. Such an approach is common during the generation of centroids for cluster searching, but would be used here to increase the efficiency with which the clusters are created in the first place.

We are currently evaluating the extent to which these two heuristic strategies affect the efficiency of clustering, and the effectiveness of searches of the resulting classifications. An alternative means of increasing the efficiency of clustering is also under consideration in our laboratory, this being the use of parallel computer hardware to carry out the NN-searches. Previous work [SALT81, FOGU84] has shown the potential of such hardware for the matching of queries with documents, and it seems possible that the approach could also be highly appropriate for the identification of RNN-pairs in clustering applications.

It seems likely that only with such software and hardware developments will it be possible to use the present algorithm to generate Ward clusterings of document collections of realistic size.

5. CONCLUSIONS

In this paper, we have presented a detailed description of the implementation of a program to obtain Ward, or minimum variance, classifications of several collections of documents. The work is based upon a new group of hierarchic clustering algorithms which make extensive use of nearest neighbour searching. We have described the use of this algorithm with an NN-searching procedure for bibliographic data, and have discussed various strategies which can be used to increase the efficiency of clustering.

We note in conclusion that although the work described here has been carried out specifically to support research into document clustering, the program might be used in the analysis of any data which is characterised by very sparse data matrices. Examples of other possible textual applications include the grouping of works upon the basis of vocabulary usage for authorship studies, the identification of new research areas upon the basis of author citation and co-citation behaviour, and the clustering of keywords for query expansion. Potential non-textual applications include the clustering of archaeological sites upon the basis of the artefacts that have been discovered, and the clustering of chemical compounds upon the basis of the presence or absence of structural or spectral features.

Acknowledgement: We thank the British Council for the award of a research studentship to AEH.

REFERENCES

[BENT80] Bentley, J.L., Weide, B.W. and Yao, A.C. (1980), "Optimal Expected-Time Algorithms for Closest Point Problems," ACM Transactions on Mathematical Software, 6, 563-580.

[CROF77] Croft, W.B. (1977), "Clustering Large Files of Documents Using the Single-Link Method," Journal of the American Society for Information Science, 28, 341-344.

[CROF80] Croft, W.B. (1980), "A Model of Cluster Searching Using Classsification," Information Systems, 5, 189-195.

[DAYE84] Day, W.H.E and Edelsbrunner, H. (1984), "Efficient Algorithms for Agglomerative Hierarchical Clustering Methods," Journal of Classification, 1, 7-24.

[DEFA77] Defays, D. (1977), "An Efficient Algorithm for a Complete Link Method," Computer Journal, 20, 93-95.

[FRIE77] Friedman, J.H., Bentley, J.L. and Finkel, R.A. (1977), "An Algorithm for Finding Best Matches in Logarithmic Expected Time," ACM Transactions on Mathematical Software, 3, 209-226.

[GOFF69] Goffman, W. (1969), "An Indirect Method of Information Retrieval," Information Storage

and Retrieval, 4, 363-373.
[GRIF84] Griffiths, A., Robinson, L.A. and Willett, P. (1984), "Hierarchic Agglomerative Clustering Methods for Automatic Document Classification," Journal of Documentation, 40, 175-205.
[GRIF86] Griffiths, A., Luckhurst, H.C. and Willett, P. (1986), "Using Inter-Document Similarity Information in Document Retrieval Systems," Journal of the American Society for Information Science, 37, 3-11.
[JARD71a] Jardine, N. and Sibson, R. (1971), Mathematical Taxonomy New York: Wiley.
[JARD71b] Jardine, N. and Van Rijsbergen, C.J. (1971), "The Use of Hierarchic Clustering in Information Retrieval," Information Storage and Retrieval, 7, 217-240.
[KITT78] Kittler, J. (1978), "A Method for Determining K-Nearest Neighbours," Kybernetes, 7, 313-315.
[LANC67] Lance, G.N. and Williams, W.T. (1967), "A General Theory of Classificatory Sorting Strategies. I. Hierarchical Systems," Computer Journal, 9, 373-380.
[MURT82] Murtagh, F. (1982), "A Very Fast, Exact Nearest Neighbour Algorithm for Use in Information Retrieval," Information Technology: Research and Development, 1, 275-283.
[MURT83] Murtagh, F. (1983), "A Survey of Recent Advances in Hierarchical Clustering Algorithms," Computer Journal, 26, 354-359.
[MURT84a] Murtagh, F. (1984), "Complexities of Hierarchic Clustering Algorithms: State of the Art," Computational Statistics Quarterly, 1, 101-113.
[MURT84b] Murtagh, F. (1984), "A Review of Fast Techniques for Nearest Neighbour Searching," in COMPSTAT 1984, Vienna: Physica-Verlag.
[NORE77] Noreault, T., Koll, M. and McGill, M.J. (1977), "Automatic Ranked Output from Boolean Searches in SIRE," Journal of the American Society for Information Science, 28, 333-339.
[PERR83] Perry, S.A. and Willett, P. (1983), "A Review of the Use of Inverted Files for Best Match Retrieval in Information Retrieval Systems," Journal of Information Science 6, 59-66.
[POGU84] Pogue, C.A. and Willett, P. (1984), "An Evaluation of Document Retrieval from Serial Files Using the ICL Distributed Array Processor," Online Review 8, 569-584.
[SALT71] Salton, G. (1971), The SMART Retrieval System, New Jersey: Prentice-Hall.
[SALT81] Salton, G. and Bergmark, D. (1981), "Parallel Computations in Information Retrieval," Lecture Notes in Computer Science, 111, 328-342.
[SALT83] Salton, G. and McGill, M.J. (1983), Introduction to Modern Information Retrieval, New York: McGraw-Hill.
[SIBS73] Sibson, R. (1973), "SLINK: an Optimally Efficient Algorithm for the Single-Link Cluster Method," Computer Journal, 16, 30-34.
[SMEA81] Smeaton, A.F. and Van Rijsbergen, C.J. (1981), "The Nearest Neighbour Problem in Information Retrieval. An Algorithm Using Upperbounds," ACM SIGIR Forum, 16, 83-87.
[VANR74] Van Rijsbergen, C.J. (1974), "Further Experiments with Hierarchic Clustering in Document Retrieval," Information Storage and Retrieval, 10, 1-14.
[VANR75] Van Rijsbergen, C.J. and Croft, W.B. (1975), "Document Clustering: an Evaluation of some Experiments with the Cranfield 1400 Collection," Information Storage and Retrieval, 11, 171-182.
[VANR79] Van Rijsbergen, C.J. (1979), Information Retrieval London: Butterworth.
[WEIS81] Weiss, S.F. (1981), "A Probabilistic Algorithm for Nearest Neighbour Searching," in Oddy, R.N. et al. (eds.) Information Retrieval Research, London: Butterworth.

	Keen	Cranfield	Evans
Number of documents	800	1400	2542
Number of index terms	1432	2557	3730
Mean number of terms/document	9.8	28.7	6.6

Table I

Details of the three document collections

P	Keen	Cranfield	Evans
2	107	171	247
3	153	261	443
4	282	539	1188
5	182	302	467
6	54	99	148
>6	21	27	48
Total	799	1399	2541

Table II

Numbers of NN-searches, P, required to identify the RNN-pairs for each collection

K	Keen	Cranfield	Evans
1	373	2141	1477
2	300	1702	1172
3	287	1428	1136
4	282	1517	1076 *
5	280	1357	1117
6	278 *	1356 *	1125
7	283	1376	1128
8	284	1464	1112
9	284	1359	1135
10	286	1359	1141

Table III

Variation in CPU run time with K, the number of nearest neighbours stored for each point

User-oriented Document Clustering: A Framework for Learning in Information Retrieval

J. S. Deogun, Department of Computer Science, University of Nebraska
Lincoln, Nebraska, U.S.A. 68588-0115

V. V. Raghavan, Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2

ABSTRACT

In information retrieval, cluster analysis is an important tool employed to enhance both efficiency and effectiveness of the retrieval process. Most clustering algorithms have difficulty in reflecting the closeness of documents as perceived by the user. A two phase scheme for document clustering, whose results reflect the "conceptual" clusters that are perceived by the user of the retrieval system, is proposed. Since the clusters obtained by this scheme are not characterized in terms of the document representations, a strategy for cluster searching is also developed. Both the proposed document clustering scheme and document searching strategy are experimentally evaluated using a test collection from the SMART system. The preliminary experimental results obtained are very encouraging.

1. INTRODUCTION

The process of clustering analysis or classification encompasses many diverse techniques for discovering structure within complex bodies of data. Such structure is expressed in the form of classes or clusters, where the objects within a class are more similar to each other than those from different classes. The notion of similarity can vary a great deal depending on the application environment, the object representation, certain judgemental factors of specialists in the field, etc. The end result of the process of classification, which is a set of classes, is (also) referred to as a classification.

Classification is employed in Information Retrieval (IR) to improve the efficiency and effectiveness of the search process. In IR systems, where the aim is the retrieval of references from bibliographic databases, the objects to be clustered are documents such as books, reports, journals, etc. From time to time, there arises a need also for the classification of index terms (keywords) or for the classification of user queries. Automatic methods proposed by Doyle [1], Dattola [2], Rocchio [3], van Rijsbergen and Croft [4], Salton and Wong [5] and Yu [6,7] are examples of algorithms for document clustering. Keyword clustering has been studied by Sparck-Jones [8], Gottlieb and Kumar [9], Dattola and Murray [10], Yu [11] and others.

In an IR environment, a collection of documents is given. There is also a group of users. The information need of a user (at a particular time) can be met by his reading one or more of the documents. That is, the system does not directly provide the answers to the question in the user's mind, but rather points the user to appropriate documents. Naturally the notion of relevance plays an important role in this context. If a document is judged by the user to be of interest, it is termed relevant; it is non-relevant otherwise. Thus, with respect to a particular need, it is assumed that there is a classification in the user's mind that separates the relevant documents from the non-relevant ones. The IR system, by whatever mechanisms available to it, must discover or characterize this dichotomy.

But this characterization of relevance is complex. A document may or may not be relevant to a user query depending on many variables concerning the document (e.g. its scope, how it is written) as well as numerous user characteristics (e.g. why the search was initiated, user's previous knowledge). Furthermore, the system of representation used to specify the content of documents and user needs is often simplistic, for reasons of efficiency. Since many such factors affect the judgment (by the system) concerning relevance in a complex way, it is recognized that an IR system cannot precisely select only and all relevant documents.

In order to cope with this environment and to facilitate reasonable retrieval performance, researchers in IR have advocated the classification of documents. The retrieval strategies based on classification of documents can lead to greater efficiency by the search being restricted to just a few selected clusters; they can achieve a greater effectiveness if clusters are formed in such a way that, when the appropriate clusters are selected for retrieval (or detailed examination), the number of relevant documents obtained relative to the number of documents retrieved (or examined) is high. This implies that the clustering algorithm must ensure that documents that are likely to be relevant to the same queries are placed together in the same clusters. With this aim in mind, a vast number of clustering algorithms have been proposed. Such algorithms can be categorized in many different ways.

One useful categorization is in terms of the emphasis placed on the computation of the pairwise similarities (or distances) between objects (documents, index terms, etc.). At the one extreme there are methods where the computation of the similarity between any pair of objects is the first step in the process of identifying clusters. That is, an object-object similarity matrix, whose (i,j)th element represents the closeness between the i-th and the j-th objects, is created. The similarity is usually expressed as a function of the lists of attributes or feature vectors that represent the objects. At the other extreme, there can be methods that do not even require that feature vectors be associated with the objects to be classified. Naturally, these methods would not require the computation of object-object similarity measures. Between the two extremes lie a number of approaches that require the computation of only selected object-object similarities and are efficient for that reason.

Jardine and Sibson [13] have identified a formal model of classification methods. In essence, the clustering schemes analyzed by Jardine and Sibson consist of first constructing the object-object similarity matrix and then obtaining a "target" similarity matrix from the object-object similarities. Finally, a classification of the objects is generated by employing the maximal complete subgraph (MCS, or clique) clustering. The specific steps involved in the formation of the target similarity matrix is the most critical element in determining the type of classification generated. Most clustering methods based on graph theory notions can be given an equivalent description within the context of Jardine and Sibson's model. For instance, the single-link clustering [4], which is quite popular in IR, can easily be described within their framework. The methods by Gottlieb and Kumar [9], Salton and Wong [5], and Dattola and Murray [10] can also be described in this manner.

However, the approaches requiring the computation of the similarity may be too slow when the number of objects to be clustered is large. Specifically, such algorithms require computing time $O(n^2)$ just for the similarity calculations, for n objects. In order to reduce the operational requirements, methods that do not require all pairwise similarities have been proposed. Doyle [1], Dattola [2], and Rocchio [3], among others, have developed such algorithms. Usually, these methods either obtain an initial solution or start with a partial solution, and modify this classification as the objects examined by the system in some sequence. Since a new object that is encountered can be compared to cluster centroids (or representatives) of existing clusters in order to decide upon its placement, all pairwise similarities are not needed. Thus, the fact that objects may be clustered by making a "single-pass" through them leads to the increased efficiency. However, these methods are order dependent; that is, the resulting classification is sensitive to the order in which the objects are selected for processing.

It is worth noting that Jardine and Sibson-type algorithms lead very naturally to multi-level classifications, whereas the single pass-type algorithms are intended to generate just one level of clusters. A method such as that proposed by Yu [7] belongs to the other extreme of the spectrum identified above. Here clusters are identified "directly" by simply determining which documents are consistently accessed together by past queries. In other words, clusters are not characterized in terms of similarities between documents defined as a function of their representation.

There are problems with both kinds of approaches. On the one hand, the methods using document-document similarities must assume that the cluster hypothesis holds. Informally, the cluster hypothesis states that "the associations (similarities) between documents convey information about the relevance of documents to requests" [14,15]. The general nature of results using such tests are quite perplexing. For one thing, the cluster hypothesis does not hold for many experimental collections when certain popular similarity functions are adopted. Moreover, even when it holds according to a given test, it may not translate into good retrieval performance [14].

On the other hand, when document clusters are generated by totally disregarding document representations, as in [7], the user's concept of which documents "go together" is captured in the classification obtained. That is, documents relevant to the same queries are made to belong to the same cluster. Since it is not possible to determine why documents with the same representation are in different clusters, the retrieval component will have a difficult time coming up with a strategy for deciding which clusters to search.

In this paper we propose an alternative to the above extremes. The central idea is to first determine documents that are consistently seen by users as belonging together; then an attempt is made to characterize these clusters by considering the power of various index terms in distinguishing documents in a cluster from those outside the cluster. The paper is organized into the following sections: In the next section, the clustering scheme we propose is motivated and described. Then, in Sec. 3, an approach for the characterization of clusters is presented. Given this characterization, a criterion for deciding the clusters to be selected with respect to new queries is also proposed. The proposed scheme is experimentally evaluated in Sec. 4. Finally, in Sec. 5, the conclusions of this investigation are presented.

2. USER-ORIENTED CLUSTERING APPROACH

In this section a clustering scheme, which captures the user's concept of closeness between documents, is developed. The aim here is essentially the same as that of Yu [7]. However, the details of the clustering scheme are different. First, the goals of and the motivation for pursuing this approach are presented. Then, the particular algorithm is described in detail.

In a user-oriented clustering approach, we desire a classification of the documents that is consistent with the behavior of the past users. In other words, if certain documents are desired together repeatedly, by many users or a number of times by some users, then those documents are to be placed into a single cluster. Several ideas that characterize this process are:

(i) User feedback is involved. The system tries to synthesize, on the basis of the feedback from many past users, document clusters such that documents in the same cluster would be perceived as being relevant to many queries. Thus, the clusters are a kind of summary of the past interaction with the system's users. It is expected that over a long period of time useful and meaningful clusters will emerge.

(ii) During the clustering phase, the system neither cares nor tries to understand why certain documents get placed in the same cluster. In other words, there is no attempt made to distinguish, using document representations, a document belonging to a cluster from another document not belonging to the cluster.

(iii) The requirement that clusters be made up of documents relevant to the same queries relates to the EFFECTIVENESS or the quality of the classification. In addition, other constraints such as the number of clusters or the size of a typical cluster, etc. may be imposed for reasons of retrieval EFFICIENCY.

(iv) Although the idea of employing user feedback resembles the notion of relevance feedback [16], quite well known in IR literature, the objective here is not to learn how to distinguish relevant documents from non-relevant ones vis-a-vis a particular query or system use. Instead, the system attempts to determine certain stable patterns of usage over many past user queries.

(v) In order to ensure that a reasonable mix of queries is presented to the system, some measure of importance can be associated with the various user queries. This can be based on empirical observations such as the 80-20 rule, which states that "20% of the transactions access 80% of the records, and so on" [17].

Yu [7] recently proposed an algorithm that adopts the principles mentioned above. The scheme, termed adaptive document clustering, performs cluster analysis directly by observing which documents are frequently accessed together by queries. The main steps in their algorithm are as follows:

1. Given documents R_1, R_2, ..., R_n, each document R_i is assigned an arbitrary position X_i on a real line (-inf.,inf.), for $1 <= i <= n$.

2. Given a (next) query
   a. use the (actually) stored clusters to provide response
   b. obtain feedback from user as to which documents are relevant to the query.
   c. modify the position of documents on the line, so that the documents accessed for this query occupy positions closer to each other on the line.

3. a. If the clustering based on the positions of the documents on the line is significantly different from the stored clusters, then reorganize the stored clusters according to the new clusters indicated.
   b. Go to step 2.

In step 2(c) above, two kinds of operations are employed. First, the set of documents relevant to the current query move toward the centroid of the corresponding positions on the line. The amount of movement of a document is proportional to its (Euclidean) distance form the centroid. Since this operation could make the positions of all the documents bunch together, a second operation that pulls certain documents away, is used. This is done by selecting k documents at random and moving those documents away from their own centroid by an amount proportional to their distance from the centroid. These operations are repeated for many queries.

The algorithm proposed in this work is motivated by the observation that the use of the real line in Yu's approach is constraining. The relative distance between objects are forced to be represented by distances between various points on the line. Furthermore, we note that, since there is a need for movement of points on the line corresponding to each instance when two documents are jointly needed, the saving achieved in computing time as a result of using the real line is somewhat unimportant. In other words, any algorithm that can generate the clusters in time proportional to the amount of input should be, computationally, equally attractive.

In the algorithm proposed in this paper, we make explicit use of a simple theoretical result. It can be shown that if retrieval efficiency is not a concern and clusters are required to be non-overlapping, then the optimum classification is one where every cluster consists of the maximal set of documents that are relevant exactly to a subset of the queries and not relevant to any query not in the subset. Thus, the algorithm consists of two phases: (A) Determine the optimum classification, according to the characterization above and (B) Determine which of these clusters should be combined to form larger clusters (taking retrieval efficiency into account). The result from A, while guaranteeing that no extraneous documents have to be selected with respect to any query, suffers from the drawback that clusters may be too small and that too many clusters have to be retrieved. In contrast, it is expected that after employing phase B some quality or effectiveness is compromised in order to bring the retrieval effort to a reasonable level.

Phase A: In this phase the document collection is divided into a number of clusters based on an initial set of queries. At the beginning of phase A, the entire document collection is assumed to form a single cluster. As first query is processed the existing cluster is divided into two clusters corresponding to relevant and non-relevant sets. Each time a new query is processed, each cluster that have non-empty intersection with the response set of this query is divided into two clusters based on relevant and non-relevant sets. The phase A continues until all queries in the initial set of queries has been processed. The clusters thus obtained, in fact correspond to minterms of a

boolean expression formed by combining all queries in the initial set of queries.

Phase B: In phase B, the clusters obtained in the phase A are combined to form larger clusters. For each cluster C_i in phase A that has yet not been combined, the value of an affinity function is computed with respect to all other clusters and the cluster C_i is combined with a cluster C_j for which the value of the affinity function is maximum. Details of an affinity function are given in Sec. 4. The choice of an affinity function, however, depends on the application on hand.

3. CHARACTERIZATION OF CLUSTERS - THE LEARNING COMPONENT

Having obtained a classification of the documents that reflects the "conceptual" clusters shared by many users, we are next interested in using the classification to retrieve documents relative to some new queries. This step is not as trivial as it may appear at first. This is because, the system does not really know why certain documents have been assigned to the same clusters, while others are placed in different clusters. Thus, given a new query, it is not known how to select clusters that are likely to be of interest to the user.

Yu [7] recognize this problem. It is suggested that a technique called pseudo-classification [11, 12, 18] might be used to determine semantic relationships between query terms and document terms. The selection of clusters for retrieval can then be accomplished by using a generalized similarity function. This suggestion was, however, not tested by Yu. Furthermore, the process of pseudo-classification is rather time consuming. As an alternative, a scheme based on the probabilistic retrieval theory [19, 20] is developed. This approach is computationally much more attractive. The main ideas are described below. For reasons of simplicity, the ideas are presented in the context of the binary independence model [19, 20].

First, the necessary notations are introduced. Let $T = \{t_1, t_2, \ldots, t_n\}$ be the set of terms used to represent documents and queries. We assume the representations are binary, indicating only the presence or absence of the terms. Given a query q, let R and R' be, respectively, the set of relevant and non-relevant documents.

Furthermore, let p_i and q_i denote the probabilities that the term t_i is present respectively in a relevant and non-relevant document. Finally, we let D_s, for s a subset of T, represent the set of documents which contain all of the terms in s and none of the terms in T - s. Then, by the independence assumptions, the probability a document is in D_s given that it is relevant and non-relevant is given, respectively, by

$$\Pr(D_s/R) = \prod_{t_i \text{ in } s} p_i \prod_{t_j \text{ in } T-s} (1 - p_j),$$

and

$$\Pr(D_s/R') = \prod_{t_i \text{ in } s} q_i \prod_{t_j \text{ in } T-s} (1 - q_j).$$

The decision rule proposed is to retrieve a document D in D_s with respect to the request q provided

$$\Pr(R/D_s) > \Pr(R'/D_s).$$

This implies, retrieve D if

$$\frac{\Pr(D_s/R)\ \Pr(R)}{\Pr(D_s)} > \frac{\Pr(D_s/R')\ \Pr(R')}{\Pr(D_s)}$$

because of Bayes' Theorem. In the context of a particular query, Pr(R) and Pr(R') may be assumed to be constant over all D_s's. Thus, in deciding a retrieval result for q, ranking documents sets according to $\Pr(D_s/R)$ is identical to the result as if they were ranked according to $\Pr(R/D_s)$. It is granted that p_i's and q_i's can be estimated through relevance feedback.

An equivalent way of realizing the ranking involves associating a weight with each query term. The, so called, relevance weight of term t_i is given by

$$w_i = \log \frac{p_i(1-q_i)}{(1-p_i)q_i}. \qquad (3.1)$$

The output rank of any document D in D_s is then based on

$$\sum_{\substack{t_i \text{ in } s \\ t_i \text{ in } q}} w_i. \qquad (3.2)$$

Our motivation in estimating w_i's is to recognize which terms play a crucial role in distinguishing documents in the relevant set from those in the non-relevant set. A positive value of w_i indicates that the term is more likely to appear in a relevant document than in a non-relevant document. The opposite holds, for negative values of w_i.

Yu et al. [21,22] and Croft [23] have considered the application of the binary independence model to cluster searching. In that context, the documents are partitioned into a set of clusters denoted $C_1, C_2, \ldots C_m$. Compared to the developments above, a cluster C_j, $1 \le j \le m$, is given the role of R. Thus, p_{ij} and q_{ij} are the probabilities that term appears in a document given that, respectively, the document is in and is not in C_j. Following similar arguments, the term weights W_{ij}'s are given by:

$$W_{ij} = \log \frac{p_{ij}\ (1-q_{ij})}{q_{ij}\ (1-p_{ij})}. \qquad (3.3)$$

W_{ij} indicates the importance of t_i in distinguishing a document in C_j from one not in C_j.

Our proposal for selecting the cluster (or clusters) with respect to a query involves the comparison of w_i's to W_{ij}'s $1 \le j \le m$. In the extreme case, if there exists some k such that $C_k = R$, then it is clear that $w_i = W_{ik}$ for all i. More generally, we propose that a scheme be adopted that determines the cluster for which the

strongly discriminating (positively or negatively) terms relative to the cluster are also the strongly discriminating terms relative to the set of documents relevant to the given query.

For example, a function which we find promising considers, for each j, the sum of W_{ij} over those values of i for which w_i is positive i.e. the function SUM = $\sum_{i|w_i>0} W_{ij}$. This is because w_i $i|w_i> 0$ positive means that the term t_i appears mostly in relevant documents. The cluster for which this function SUM gives the largest value is deemed the most promising one to examine.

4. EXPERIMENTAL EVALUATION

Experimental results are provided to assess the effectiveness of the proposed clustering scheme. The proposal for selecting the clusters for retrieval purposes has also been implemented and experimental results are provided. For the purposes of the evaluation of retrieval performance, once a cluster is deemed "promising", it is assumed that all the documents from the cluster are retrieved. In order to make the evaluation of the clustering scheme, the determination of which clusters are promising is achieved by looking at upper bound results. In other words, it is assumed that by some magical scheme the clusters that will contain the most number of relevant documents can be selected. The results show the quality of retrieval when the best cluster is retrieved as well as when the top two clusters are retrieved.

For these experiments a document collection, which has been frequently used in SMART system environment, is employed. This collection is known as ADINUL. ADINUL is a collection of 82 documents and 35 queries in the field of documentation. Experimental results on CRN4NUL collection on the subject of thermo-dynamics are under progress and will be reported later. In the following we describe certain additional details that must be prescribed vis-a-vis phase B outlined in Sec. 2.

In phase B, the queries from phase A are repeated according to some probability distribution. This probability distribution may be based on the "80-20 rule", zipf's distribution or a distribution in which the probability of a query is inversely proportional to the size of its response set [17]. These distributions ensure that the way in which queries access various documents conforms to some reasonable pattern. For the experiments reported in this paper we follow the third option. For ADINUL collection a set of 400 queries is generated by repeating the 35 queries following a probability distribution such that probability of repeating a query is inversely proportional to the size of its response set. A weighted document graph with documents as vertices is generated for the experiment. Initially, weight of each edge in the document graph is equal to zero and as a new query is processed one is added to the weight of the edge (D_i,D_j) for each pair of documents D_i and D_j which co-occur in the response set of that query. This document graph is used to define an affinity function F between two clusters. For the purpose of defining an affinity function a cluster is considered as a subset of vertices in the document graph. For each cluster C_i in the phase A that has not yet been combined with any other cluster, the affinity function is computed with respect to every other cluster C_j in the collection as follows:

$$F(C_i,C_j) = W \times \frac{E_1}{E_2}$$

where

W = total weight of all edges between C_i and C_j,

E_1 = no. of vertices in C_i that are adjacent to vertices in C_j, and

E_2 = total no. of vertices in C_j.

Then a new cluster C_i is formed by combining C_i with C_k (i.e., $C_i = C_i \cup C_k$) such that

$$F(C_i,C_k) = \max_j F(C_i,C_j).$$

The new cluster C_i is retained in the collection of clusters from phase A, but cluster C_k is dropped. This process is continued until all clusters have been considered.

Several variations for the affinity function described above are possible. For example, a threshhold can be applied for combining two clusters. Moreover, in the experiments reported here we have not yet applied "80-20 rule" or zipf's distribution. Experiments following these and other options are under progress and detailed comparative results will be reported in a later paper. The experimental results obtained for ADINUL collection are given in Table 4.1.

For this collection, assuming that the cluster having the most number of relevant documents is retrieved for each query, the recall and precision achieved are respectively 0.7624 and 0.6255. When the best (as explained) two clusters are retrieved, the recall is 0.9446 and the precision is 0.5136. These figures give an indication of the best retrieval performance achievable given the clusters generated. Compared to traditional approaches for retrieval, it is easily seen that large values of precision are obtainable even when recall values are extremely high.

The experimental results using the retrieval scheme proposed in Sec. 3 show that, when the best cluster is retrieved, the recall and precision values are 0.3566 and 0.4355; the same figures are 0.4956 and 0.2424 when the best two clusters are retrieval. These results are reasonable in the sense that these precision values are somewhat higher, for the corresponding recall levels, than those achieved by standard retrieval methods (for example, full search using cosine similarity). Clearly, more research is needed since these results fall quite short of the upperbound results given above.

5. CONCLUSION

A scheme for document clustering, whose results reflect the "conceptual" clusters that are perceived by the users of the retrieval system, is proposed. The scheme consists of two phases: first clusters that would be optimal, in the sense that any response set to a query is precisely the union of one or more clusters, are generated; secondly, based on frequency with which such

clusters are jointly needed in response to queries, they are selectively combined. Experiments are performed to assess the effectiveness and efficiency of the clustering scheme. Since the clusters obtained are not characterized in terms of the document representations, a strategy for cluster searching must be devised. A proposal for this step is also described and experimental results are presented. The preliminary results that pertain to the clustering as well as the cluster searching strategies are very encouraging.

REFERENCES

1. Doyle, L.B. (1966). Breaking the Cost Barrier in Automatic Classification. Report No. SP-2516, System Development Corp., Santa Monica, Ca.

2. Dattola, R.T. (1971). Experiments with a fast algorithm for automatic classification. In the SMART Retrieval System - Experiments in Automatic Document Processing, G. Salton, Ed., Prentice-Hall, Englewood Cliffs, N.J.

3. Rocchio, J.J. Jr. (1966). Document retrieval systems - optimization and evaluation. Doctoral thesis. In: Information Storage and Retrieval, Scientific Report NO. ISR-10, Harvard Univ., Cambridge, Mass.

4. van Rijsbergen, C.J. and Croft, W.B. (1975). Document clustering: An evaluation of some experiments with the Cranfield 1400 collection. Inf. Process Manage., 11, 171-182.

5. Salton, g. and Wong, A. (1978). Generation and search of clustered files. ACM Trans. Database Systems, 3, 321-346.

6. Yu, C.T. (1974). A clustering algorithm based on user queries. J. Amer. Soc. Inform. Sci., 25, 218-226.

7. Yu, C.T. (1985). Adaptive document clustering. Proc. Eighth Annual International ACM SIGIR Conf. on Research and Development in Information Retrieval, Montreal, Canada, 197-203.

8. Sparck-Jones, K. (1971). Automatic Keyword Classification for Information Retrieval. Archon Books, Connecticut.

9. Gotlieb, C.C. and Kumar, S. (1968). Semantic clustering of index terms. JACM, 15, 493-513.

10. Dattola, R.T. and Murray, D.M. (1967). An experiment in automatic thesaurus construction. In: Information Storage and Retrieval, Scientific Report No. ISR-13, Chpt. VIII, Cornell University, Ithaca, N.Y.

11. Yu, C.T. (1975). A formal construction of term classes. JACM,22,17-37.

12. Raghavan, V.V. and Yu, C.T. (1979). Experiments in the determination of the relationships between terms. ACM Trans. Database Systems., 4, 240-260.

13. Jardine, N. and Sibson, R. (1968). A model for taxonomy. Mathematical Biosciences, 2, 456-482.

14. Voorhees, E.M. (1985). The cluster hypothesis revisited. Proc. Eighth Annual International Conference on Research and Development in Information Retrieval, Montreal, Canada, 188-196.

15. Jardine, N. and van Rijsbergen, C.J. (1971). The use of hierarchic clustering in information retrieval. Information Storage and Retrieval, 7,217-240.

16. Rocchio, Jr., J.J. (1971). Relevance feedback in information retrieval. in the Smart Retrieval System- Experiments in Automatic Document Processing, G. Salton, Ed., Prentice Hall Inc., Englewood Cliffs, N.J.

17. Knuth, D.E. (1973). Sorting and Searching- The Art of Computer Programming. Vol. 3, Addison-Wesley Publishing Co., Reading, MA.

18. Salton, G. (1980). Automatic term class construction using relevance -A summary of work in automatic pseudo-classification. Inf. Process. and Manage., 16,1-15.

19. Robertson, S.E. and Sparck Jones, K. (1976). Relevance weighting of search terms. J. of Amer. Soc. Inform. Sci., 27, 129-146.

20. Yu, C.T. and Salton, G. (1976). Precision weighting-an effective automatic indexing method. JACM, 23, 76-88.

21. Yu, C.T., and Luk, W.S. (1977). Analysis of effectiveness of retrieval in clustered files. JACM,24,607-622.

22. Yu, C.T., Luk, W.S., and Siu, M.K. (1978). On the estimation of the number of records with respect to a query. ACM Trans. Database Systems,3,41-56.

23. Croft, W.B. (1980). A model of cluster searching based on classification. Information Systems, 5,189-195.

Average size of response set = 4.0

	after phase A	after phase B
Number of clusters	62	20
Average cluster size	1.32	4.1

Retrieval Scheme: For each query retrieve one cluster with having the greatest number of relevant documents

Recall = .7624
Precision = .6255

Retrieval Scheme: For each query retrieve two clusters with the maximum total number of relevant documents

Recall = .9446
Precision = .5136

Average number of clusters that must be retrieved to get all relevant documents = 1.81

Retrieval Scheme: Best cluster is retrieved according to function SUM described in Sec. 3.

Recall = .3566
Precision = .4355

Retrieval Scheme: Best two clusters are retrieved according to function SUM.

Recall = .4956
Precision = .2424

TABLE 4.1 SUMMARY OF EXPERIMENTAL RESULTS FOR ADINUL COLLECTION

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

The Efficiency of Inverted Index and Cluster Searches

Ellen M. Voorhees

Cornell University

Abstract: The processing time and disk space requirements of an inverted index and top-down cluster search are compared. The cluster search is shown to use both more time and more disk space, mostly due to the large number of cluster centroids needed by the search. When shorter centroids are used, the efficiency of the cluster search improves, but the inverted index search remains more efficient.

1. Introduction

Research by Griffiths *et al.* suggests that the single link clustering method is not as effective as other agglomerative, hierarchic clustering methods for document retrieval [GRIF84, GRIF85]. Recent research by the author corroborates these findings and also shows that the partial ranking produced by a top-down search of the complete link hierarchy can be more effective than ranking the collection by decreasing similarity to the query. Furthermore, the evidence suggests that searches of the complete link hierarchy will remain effective independent of collection size [VOOR86].

However, in order to be useful a search must be efficient as well as effective. This paper compares the efficiency of the cluster search to the efficiency of an inverted index search that ranks the entire collection. The efficiency of each search is measured in two ways: the number of bytes required to store the auxiliary files required by the search, and the mean time required to retrieve a set of documents for a query. The processing time is further divided into CPU time and I/O time. The measurements are obtained by observing searches on four test collections.

The results of the experiments indicate that the inverted file search is more efficient in terms of both space and processing time since the cluster search requires many centroid vectors. The maximum length of a centroid vector can be significantly decreased before the effectiveness of the cluster search suffers, but the efficiency increase gained in this way is not sufficient to make the cluster search as efficient as the inverted index search.

2. The Searches

The purpose of the inverted index and cluster searches is to return a set of documents to the user in response to a query. The documents returned are the documents encountered in the search that have the highest similarities to the query. In the current experiments, the documents and queries are represented by weighted vectors and the inner product similarity is used. The weight of a term in a vector is proportional to the number of times the term appears in the document or query and inversely proportional to the number of documents in which it appears. The weights in each vector are normalized so that the inner product similarity of two vectors is equal to their cosine similarity.

2.1. The Inverted Index Search

The inverted index search encounters all documents with non-zero similarities to the query, and thus is guaranteed to return the documents with the highest similarities. The search requires an inverted index – a representation of the collection in which access to the collection is made through terms instead

of through documents. Associated with each term in the collection is a list of <document id,weight> pairs for each document in which that term appears. The inverted index search manipulates the inverted lists of query terms as described below.

The search used here is the 'basic algorithm' described by Buckley and Lewit [BUCK85]. The search requires enough internal memory space so that a partial similarity can be accumulated between the query and each document (*e.g.* 4N bytes where N is the size of the collection). Another array contains the best *x* partial similarities seen so far (where *x* is the number of documents to be returned to the user). The search begins by setting each of the partial similarities to zero. The inverted index list for each query term is read in turn. For each document, *d*, on a list, the product of the weights of the current term in the query and in *d* is added to *d*'s partial sum. If *d*'s current similarity is large enough, it is placed in the top similarities array. After all query term lists have been processed, the top similarity array contains the documents to be returned to the user. Figure 1 shows a sample search for a query consisting of terms A, B, and C when two documents are to be returned.

2.2. The Cluster Search

Cluster searches were introduced as both a means of improving the efficiency [SMAR71] and the effectiveness [JARD71] of a retrieval system. Cluster searches do not encounter all documents, and therefore, they are not guaranteed to retrieve the documents with the greatest similarities to the query. The particular search used here requires several auxiliary files: the representation of the cluster hierarchy, the centroid file, and the document file.

The hierarchy used in these experiments is the complete link hierarchy. This hierarchy is formed by repeatedly merging the clusters with the maximum similarity, where the similarity between clusters is defined to be the minimum similarity between any pair of documents in the clusters. Due to the strict clustering criterion, complete link hierarchies tend to be quite shallow and bushy. The content of each cluster is summarized by a centroid vector; the similarity between a query and a cluster is computed using its centroid. The centroids in these experiments were created as follows:

- The sum of the within document frequency of each term in the cluster is computed, and the terms are sorted by decreasing frequency.
- The top 250 terms are selected to be in the centroid. The weight of each of the terms in the centroid is the rank (from the bottom) of the term in the sorted list – equal frequencies are assigned the same rank.
- The rank weights are multiplied by an inverse document frequency factor and normalized so the sum of the squares of the weights equals one (cosine normalization).

The number of clusters that are encountered in the search depends on the parameter *NumWanted*. This parameter specifies the number of documents that should be retrieved by the search. In general, retrieving more documents will be more effective but also more expensive than retrieving fewer documents.

The search begins by placing the root of the hierarchy into an empty heap. The top of the heap (corresponding to the cluster with the current largest similarity with the query) is popped until sufficient documents have been retrieved or the heap is empty. If the hierarchy node popped from the heap represents a document, that document is retrieved. Otherwise, all the children of the node that have non-zero similarities with the query are added to the heap. When sufficient documents have been retrieved, the retrieved set is ranked by decreasing similarity.

Since the path down the tree is controlled by the nodes that are popped from the heap, the search is neither a true depth-first nor a true breadth-first search of the hierarchy. In practice, it is closer to a depth-first search with some backtracking than to a breadth-first search.

A sample top-down search for a hypothetical hierarchy and query is given in Figure 2. In the Figure, the squares represent documents and the circles represent non-singleton clusters. The numbers within the circles are node numbers; the documents are identified by letters. The number in parentheses next to a node is the similarity of that node with the query.

3. Storage Requirements

This section investigates the amount of external memory space required to store a document collection and the data structures needed to search it. (A disk is

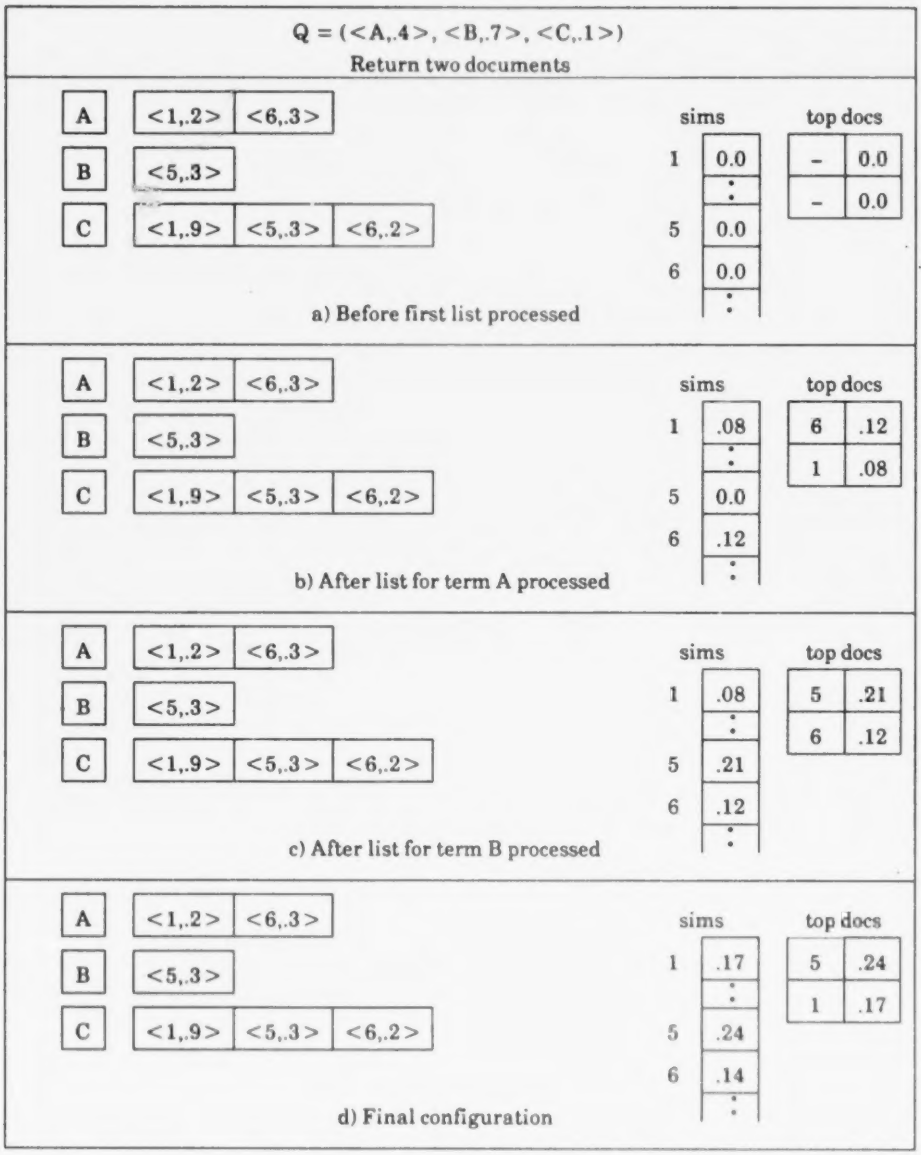
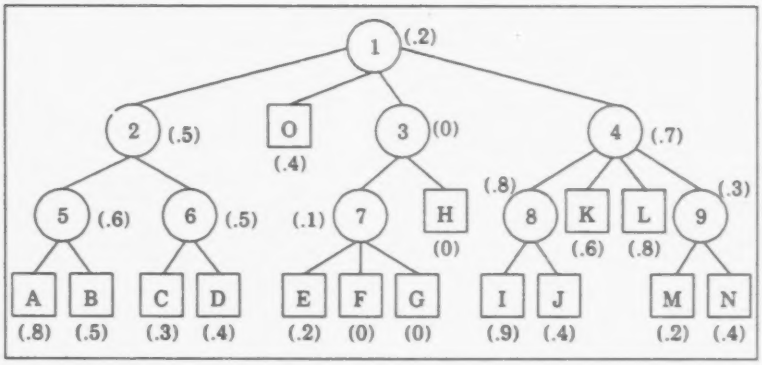


Figure 1: Sample Inverted Index Search

the secondary storage device assumed in the rest of the paper.) The size is measured in the number of bytes required to store the data assuming the data structures are implemented as described below. The storage required for the natural language text of the documents is not included in the comparisons since the same amount of storage is used for the text regardless of the searching strategy.

A vector, either a document vector or a centroid vector, consists of a fixed length header and an arbitrary number of tuples. The header contains an id number and the number of tuples in the vector



a) Hierarchy to be searched

add [1,.2]	heap: 1
pop [1,.2]	
add [2, .5], [4, .7], [O, .4]	heap: 2, 4, O
pop [4, .7]	
add [8, .8], [9, .3], [L, .8], [K, .6]	heap: 2, O, 8, 9, L, K
pop [L, .8]	1 document retrieved
pop [8, .8]	
add [I, .9], [J, .4]	heap: 2,O,9,K, I, J
pop [I, .9]	2 documents retrieved
pop [K, .6]	3 documents retrieved
	STOP
	Retrieved Documents:
	I sim = .9
	L sim = .8
	K sim = .6

b) trace of search with $NumWanted = 3$

Figure 2: Sample Top-down Cluster Search

(8 bytes). Each tuple contains a concept number and the weight of that concept in the vector (8 bytes).

An inverted index entry has a similar structure. The header contains the concept number, the number of documents in which the term appears, and a pointer to tuples that make up the rest of the entry (12 bytes). Each tuple contains the vector id and the weight of the concept in that vector (8 bytes). There is an inverted index entry for each distinct concept in the collection.

The hierarchy is described by a set of fixed length nodes, one node for each cluster (including singleton clusters) in the hierarchy. A node contains six fields: the node id, the similarity value at which the cluster it represents was formed, a pointer to the vector associated with the node, the number of children, the node id of its first child, and the node id of its rightmost sibling (24 bytes).

3.1. Disk Usage Experiments

The inverted index search needs to access only the inverted file in order to complete a search. However, to perform some other retrieval functions such as query expansion for relevance feedback, a direct file of the document vectors is also needed. The space meas-

Table 1: Collection Characteristics

	CACM	CISI	INSPEC	MED
number docs	3204	1460	12684	1033
number terms	8503	4941	14573	6927
mean terms per doc	22.5	43.9	32.5	51.6
number queries	50	35	77	30
mean relevant docs per query	15.8	49.8	33.0	23.2
total relevant docs	792	1742	2543	696
exhaustive rel. assessments	no	yes	no	yes

ured for the inverted index search, therefore, is the amount of space used by both the direct and inverted files. The space measured for the cluster search is the amount of space required by the hierarchy file, the centroid file, and the direct document file. The maximum length of a centroid is restricted to 250 terms in these experiments.

The auxiliary files needed for the inverted index and cluster searches were created for four test collections: CACM, a computer science collection; CISI, a library science collection; INSPEC, an electrical engineering collection used by permission of The Institution of Electrical Engineers, Hertfordshire, England; and MED, a bio-medicine collection. Details of these collections can be found in Table 1. (The CACM and CISI collections contain information other than standard content indicators [FOX83]. However, only the standard content indicators are used in these experiments. As a result, two queries have been removed from the CACM query collection since they contain only non-content indicators.) The size of each auxiliary file and the total size for each search is given in Table 2. The percentage increase of the cluster search over the inverted index search is also included in the Table.

The numbers in Table 2 show that the cluster search requires a significant amount more disk space than the inverted index search (an increase of 70-100%). The cluster search requires larger amounts of disk space due to the number of centroids that need to be stored – the centroid file is larger than the inverted index for each of the collections.

Secondary storage is relatively inexpensive; a more critical factor of the efficiency of a search is the amount of time it takes to retrieve a set of documents. This topic is discussed in the next section.

4. Processing Time Requirements

As was mentioned in the introduction, the total time to process a query is assumed to be the sum of the CPU time and the I/O time. The CPU time is the time the main processor devotes to retrieving documents for the query. The I/O time is the amount of time the search spends waiting for pages to be retrieved from the disk.

4.1. CPU Time

The CPU time is measured by the number of idealized machine instructions performed in the search. Each (floating point) addition, (floating point) multiplication, comparison, pointer increment, and array indexing is counted as one operation. The instruc-

Table 2: External Storage Space in Bytes for Inverted Index and Cluster Searches
(including % increase of the cluster search over the inverted index search)

	doc vectors	inverted index	total for inverted	hierarchy file	centroid file	total for clusters	% increase
CACM	603,488	679,892	1,283,380	142,392	1,481,320	2,227,200	74
CISI	539,584	572,084	1,111,668	67,320	1,375,536	1,921,850	73
INSPEC	3,399,512	3,472,916	6,872,428	589,488	9,685,704	13,674,704	99
MED	434,656	509,516	944,172	47,736	1,132,544	1,614,936	71

```
for (each query term list) {
    if (at end of list) {                                   (1 instruction)
        go to next list;
    }
    sims[current did] + = query term wt * doc term wt;      (3 instructions)
    increase list ptr to point to next did in list;         (1 instruction)
}
```

a) inverted index search

```
sim = 0;
while (true) {
    if (query concept num > doc concept number) {           (1 instruction)
        increase doc ptr;                                   (1 instruction)
        if (at end of doc) {                                (1 instruction)
            exit loop;
        }
    }
    else if (query concept num < doc concept number) {      (1 instruction)
        increase query ptr;                                 (1 instruction)
        if (at end of query) {                              (1 instruction)
            exit loop;
        }
    }
    else {
        sim + = query term wt * doc term wt;                (2 instructions)
        increase query ptr;                                 (1 instruction)
        if (at end of query) {                              (1 instruction)
            exit loop;
        }
        increase doc ptr;                                   (1 instruction)
        if (at end of doc) {                                (1 instruction)
            exit loop;
        }
    }
}
```

b) cluster search

Figure 3: Idealized Instruction Counts for Both Searches

tions are counted only in the main part of each search (*i.e.* no initialization costs are included). For the inverted index search, Figure 3(a), the main loop is the loop which processes each inverted list. Five machine instructions are required for each document id in the lists. For the top-down search, Figure 3(b), the main part is computing the similarity between a given query and document or centroid vector. The cluster search performs three instructions for each term that appears in exactly one of the query and centroid (or document) vectors and six instructions when a match occurs.

4.2. I/O Time

The amount of information that needs to reside in core at any given time is not very large for any of the searches. Assuming the inverted lists contain the

weights of the terms in the documents, the inverted index search requires only the inverted lists of each query term. The top-down search needs to keep the hierarchy nodes that have been visited and the current centroid or document vector in core. Each vector is needed only once (if it is needed at all) so the same memory location can be used for each vector. The hierarchy nodes may be accessed more than once, so they must be stored in separate locations. Since each node consists of only 24 bytes and the top-down search is closer to a depth-first search than a breadth-first search, the amount of space required by the hierarchy nodes is not excessive. For example, the mean number of nodes accessed in the top-down search of the complete link hierarchy for the INSPEC collection is 873 (requiring only 20,952 bytes of core). These memory usage considerations do not include the memory needed by the program variables such as the array of similarities for the inverted index search and the heap used to store the similarities in the cluster search.

Any modern computer will have enough memory to meet these demands, and thus no paging strategy is considered in the experiments. That is, it is assumed that if a page that has already been brought into main memory for this query is accessed again, the page still resides in core. The I/O time of a search, therefore, is proportional to the number of data pages that are accessed at least once, and this figure is used as the measure of the I/O cost of a search. This measure makes the unrealistic assumption that each page access takes the same amount of time. In fact, the time necessary to access a page is the sum of the seek time, the rotational delay, and the data transfer times, so some page accesses are less expensive than others. However, the experiments do not use this level of detail.

Another implicit assumption in this measure is that the cost of presenting the same number of documents as output is approximately equal for the two searches. This assumption is also untrue: the natural language text for the top-down search can be stored in cluster order while the text for the inverted index search is stored randomly. Clustering the text should cause fewer disk accesses to be required since documents on the same page are likely to be retrieved together. Furthermore, the cost of accessing sequential pages is less than accessing the same number of random pages. However, due to the small number of documents retrieved, this difference is negligible compared to the difference in the number of disk accesses required in the searches themselves.

In order to count the number of pages accessed during a search, some allocation of the data to pages must be assumed. The model of page allocation used in these experiments is a compromise between making the searches as efficient as possible and accurately modeling an operational environment. Collection maintenance – retiring old documents and adding new documents – is an important consideration in an operational environment. Thus the model cannot assume the pages are allocated in such a way that the efficiency would seriously deteriorate or the entire collection would need to be reconfigured if new documents are added to the collection.

The page length is assumed to be 4096 bytes. Inverted index headers are packed into as few pages as possible without splitting a header across page boundaries. Each list starts on a new page and takes as few pages as possible without splitting a tuple across page boundaries. Starting each list on a new page allows the index to be easily enlarged when documents are added to the collection, and is more representative of larger collections where each query term is likely to require a page access. Hierarchy nodes that are siblings are stored together beginning on a new page; they take as few pages as possible without splitting a node across page boundaries. The vectors associated with sibling nodes are also packed into as few pages as possible. Having only siblings on a page (as opposed to packing hierarchy nodes as tightly as possible) allows the hierarchy to be easily updated when documents are added to or removed from the collection. It also makes the model more realistic of larger collections since this implementation renders back-tracking more expensive than it would be with a packed representation.

4.3. Processing Time Experiments

The efficiency statistics for the searches are presented in Table 3. The statistics given are the mean number of idealized machine instructions performed and the mean number of pages accessed for the query set of each collection assuming each query is

Table 3: Efficiency Statistics for Inverted Index and Cluster Searches

collection	inverted index search			cluster search		
	mean instructions	mean # pages	mean seconds	mean instructions	mean # pages	mean seconds
CACM	9,296	20.3	.62	129,230	155.1	4.78
CISI	8,339	15.1	.46	105,583	158.2	4.85
INSPEC	67,877	48.9	1.54	517,999	418.3	13.07
MED	3,136	18.1	.55	71,632	106.5	3.27

independent of all other queries and twenty documents are to be retrieved. Table 3 also includes the mean number of seconds to process a query assuming a page access takes .03 seconds (approximately 30 pages per second) and a machine instruction takes 1 microsecond.

The inverted index search is clearly more efficient than the cluster search: the cluster search performs 663% (INSPEC) to 2184% (MED) more machine instructions and accesses 488% (MED) to 948% (CISI) more pages than the inverted index search. The difference is caused by the fact that the inverted index search takes full advantage of the sparsity of the document vectors. In the inverted index search, only the terms that appear in *both* the document (or centroid) and the query are used. In the cluster search, a large percentage of the documents are avoided, but the centroids and documents that are encountered are exhaustively examined. Note in Figure 3(b) a term that occurs in only one of the query or document vector causes three machine operations but contributes nothing to the similarity; terms that occur in both vectors cause only six machine instructions. This problem is exacerbated in the search of the complete link hierarchy since that hierarchy is so bushy – every search begins by examining many long centroids.

5. Shorter Centroids

Both the extra disk space and the extra processing time needed by the cluster search have been attributed to centroids. Clearly if shorter centroids were used the efficiency of the search would improve, although the effectiveness might suffer. This section investigates the effect of decreasing the maximum length of centroid vectors.

5.1. Effectiveness

The cluster search was repeated on each collection using centroids of both maximum length 100 and maximum length 75. The effectiveness of the searches is summarized in Table 4. Three different statistics are given for each search: the mean precision after ten (top number) and twenty (bottom number) documents are retrieved, the total number of relevant documents retrieved by all queries after ten and twenty documents are retrieved, and the total number of queries

Table 4: Effectiveness of Inverted Index and Cluster Searches

collection	inverted index search			cluster search centroids of length 75			cluster search centroids of length 100			cluster search centroids of length 250		
	mean prec	total # rel. ret.	# q's	mean prec	total # rel. ret.	# q's	mean prec	total # rel. ret.	# q's	mean prec	total # rel. ret.	# q's
CACM	.230 .185	115 185	9 5	.260 .215	130 215	8 7	.252 .213	126 213	9 7	.248 .199	124 199	8 6
CISI	.269 .260	94 182	7 5	.291 .241	102 169	9 6	.309 .253	108 177	7 3	.274 .237	96 166	6 3
INSPEC	.384 .315	296 485	3 2	.387 .308	298 475	10 5	.397 .314	306 484	7 4	.373 .284	287 437	6 4
MED	.620 .528	186 317	0 0	.660 .600	198 360	0 0	.667 .600	200 360	0 0	.650 .603	195 362	0 0

Table 5: Cluster Search Sizes for Varying Maximum Centroid Lengths

	centroid file size in bytes			% decrease in centroid size			% increase in total size over inverted index search		
	75	100	250	100/250	75/250	75/100	75	100	250
CACM	978,376	1,136,896	1,481,320	23	34	14	34	47	74
CISI	766,328	936,240	1,375,536	32	44	18	23	39	73
INSPEC	6,110.384	7,160,584	9,685,704	26	37	15	47	62	99
MED	564,864	711,512	1,132,544	37	50	21	11	26	71

that retrieved no relevant documents after ten and twenty documents are retrieved.

For all centroid lengths, the cluster search is more effective than the inverted index search after ten documents are retrieved; in fact, the mean precision of the cluster search using centroids of maximum length 100 is greater than that of the search using centroids of maximum length 250. Note, however, that more queries retrieved no relevant documents in the search that used centroids of maximum length 100. In general, the search that used centroids of maximum length 75 had more queries that retrieved no relevant documents than any other search, and its mean precision was worse than that of the search in which centroids were restricted to a maximum length of 100.

5.2. Disk Space

The disk space in bytes used by the cluster search is given in Table 5; also included in the Table are the percentage decrease in the size of shorter centroids to the size of longer centroids, and the percentage increase of the total size for the cluster search to the total size for the inverted index search. Although there is some decrease in the required space when the centroids are shortened, the decrease is not large enough to make any centroid file smaller than the inverted file of the document collection (see Table 2). For the large INSPEC collection, the total size needed by the cluster search with centroids of maximum length 75 is still 47% larger than the total size needed by the inverted index search.

Note that the space required by the centroids does not decrease as much as might be expected when the maximum centroid length is reduced to 100 from 250. Indeed, a decrease of at most 37% is observed in the size of the centroid file (whereas the maximum centroid length was reduced by more than half). This minimum amount of reduction can be attributed to the shape of the complete link hierarchy: since the hierarchy contains small, tight clusters, some clusters do not contain more than 100 distinct terms. More clusters contain at least 75 terms, so the decrease in size from a maximum length of 100 to 75 terms is proportionally larger.

5.3. Processing Time

Although some clusters do not contain 100 terms, top-level clusters are the most likely clusters to contain many terms since they are the largest clusters in the hierarchy. Thus in spite of the relatively small reduction in size, significant processing time may be saved by using shorter centroids in the cluster search.

The processing time results are given in Table 6. The inverted index search is still the most time efficient search. Unsurprisingly, using shorter centroids is always more efficient than using longer centroids. Somewhat more surprisingly, only the INSPEC collection exhibits a large decrease in the number of pages accessed. An explanation for this behavior can be found in Table 7 which contains the mean number of pages accessed to retrieve hierarchy nodes and the mean number of pages accessed to retrieve vectors. When shorter vectors are used, the number of pages accessed to retrieve vectors decreases but the number of pages accessed to retrieve hierarchy nodes increases. It is harder to distinguish among clusters when shorter centroids are used, and thus the search with shorter centroids is broader than the search with longer centroids. The slight increase in the number of pages accessed due to backtracking is negligible only for the large INSPEC collection.

Table 6: Processing Time of Inverted Index and Cluster Searches

collection	inverted index search			cluster search centroids of length 75			cluster search centroids of length 100			cluster search centroids of length 250		
	mean instrs	mean pages	secs	mean instrs	mean pages	secs	mean instrs	mean pages	secs	mean instrs	mean pages	secs
CACM	9,296	20.3	0.62	78,524	133.6	4.09	91,186	138.7	4.25	129,230	155.1	4.78
CISI	8,339	15.1	0.46	50,568	148.1	4.49	61,021	147.3	4.48	105,583	158.2	4.85
INSPEC	67,877	48.9	1.54	221,141	218.1	6.76	276,867	257.2	7.99	517,999	418.3	13.07
MED	3,136	18.1	0.55	32,109	87.1	2.65	39,581	90.9	2.77	71,632	106.5	3.27

Table 7: Mean Page Accesses to Retrieve Hierarchy Nodes and Vectors in Cluster Search

collection	centroids of length 75		centroids of length 100		centroids of length 250	
	hierarchy	vectors	hierarchy	vectors	hierarchy	vectors
CACM	49.0	84.3	47.4	91.4	43.6	114.7
CISI	66.6	81.6	63.2	84.2	56.7	105.7
INSPEC	53.0	165.0	54.6	202.6	41.5	354.5
MED	38.5	48.5	38.5	52.5	35.3	69.3

6. Summary

In earlier work, a top-down search of the complete link hierarchy was shown to be more effective than an inverted index search [GRIF84, VOOR86]. Unfortunately, the results of this study show the cluster search is much more expensive than the inverted index search in terms of both disk space and processing time. Decreasing the maximum length of centroids increases the efficiency of the cluster search, especially for large collections, and the centroids can be made quite short before the effectiveness of the cluster search suffers. However, the centroid length cannot be reduced enough to make the cluster search as efficient as the inverted index search while remaining as effective as the inverted index search.

It should be stressed that these experiments try only one method of increasing the efficiency of the cluster searches. Other ways of increasing the efficiency of the cluster search without sacrificing its effectiveness should be investigated. For example, inverting the top-level centroids of the hierarchy and starting the cluster search by doing an inverted search on those centroids should save time and would not increase the disk storage (since the inverted centroids could be removed from the direct centroid file). Furthermore, the effectiveness would remain the same since the same clusters would be explored.

Similarly, it may be possible to modify the complete link hierarchy to make searching it easier. An obvious possibility is to cluster the top-level centroids using some other clustering method and thus add several new levels to the hierarchy. Such a hybrid hierarchy would be more time efficient to search since fewer centroids would be at the new top level, but the same effectiveness would no longer be guaranteed.

Cluster searching also affords the possibility of storing the natural language text of a document with its representative in the cluster hierarchy. This may increase the efficiency of the cluster search by avoiding additional page accesses when displaying the retrieved documents to the user. This enhancement is not possible with the inverted index search since the terms of a document are usually spread throughout the index.

Acknowledgments

Thanks go to Professor Gerard Salton and the members of the SMART group at Cornell for many helpful discussions, and especially for suggestions on how to improve the efficiency of the cluster searches.

References

[BUCK85] Buckley, C. A., Lewit, A. F., "Optimization of Inverted Vector Searches", *Proceedings of the Eighth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 97-110, 1985. ACM order number 606850.

[FOX83] Fox, E. A., "Characterization of Two New Experimental Collections in Computer and Information Science Containing Textual and Bibliographic Concepts", Technical report 83-560, Department of Computer Science, Cornell University, September 1983.

[GRIF84] Griffiths, A., Robinson, L. A., Willett, P., "Hierarchic Agglomerative Clustering Methods for Automatic Document Classification", *The Journal of Documentation* XL(3):175-205, 1984.

[GRIF85] Griffiths, A., Luckhurst, H. C., Willett, P., "Using Interdocument Similarity Information in Document Retrieval Systems", *Journal of the American Society for Information Science* XXXVII(1):3-11, 1986.

[JARD71] Jardine, N., van Rijsbergen, C. J., "The Use of Hierarchic Clustering in Information Retrieval", *Information Storage and Retrieval* VII(5):217-240, 1971.

[SMAR71] Salton, G., editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, 1971.

[VOOR86] Voorhees, E. M., *The Effectiveness and Efficiency of Agglomerative Hierarchic Clustering in Document Retrieval*, PhD thesis, Cornell University, 1986.

ON EXTENDING THE VECTOR SPACE MODEL FOR BOOLEAN QUERY PROCESSING *

S. K. M. Wong
W. Ziarko
V. V. Raghavan
and
P. C. N. Wong

Department of Computer Science University of Regina
Regina, Canada S4S 0A2

Abstract. An information retrieval model, named the Generalized Vector Space Model (GVSM), is extended to handle situations where queries are specified as (extended) Boolean expressions. It is shown that this unified model, unlike currently available alternatives, has the advantage of incorporating term correlations into the retrieval process. The query language extension is attractive in the sense that most of the algebraic properties of the strict Boolean language are still preserved. Although the experimental results for extended Boolean retrieval are not always better than the vector processing method, the developments here are significant in facilitating commercially available retrieval systems to benefit from the vector based methods. The proposed scheme is compared to the p-norm model advanced by Salton and coworkers. An important conclusion is that it is desirable to investigate further extensions that can offer the benefits of both proposals.

1. INTRODUCTION

Information Retrieval (IR) is a discipline involved with the organization, structuring, analysis, storage, searching and dissemination of information. IR systems are designed to make available a given stored collection of information items with the objective of providing, in response to a user query, references that would contain the information desired by the user. In other words, the system is intended to identify which documents the user should read in order to satisfy his (her) information requirements.

In this environment, we have a collection of documents (e.g. books, journal, articles, technical reports, etc.). In order to identify which documents the user should read with respect to his information requirements, some method for the representation of what the documents are about (i.e. knowledge representation of documents) is needed.

A document may or may not be relevant to a user query depending on many variables concerning the document (e.g. its scope, how it is written) as well as numerous user characteristics (e.g. why the search is initiated, user's previous knowledge). In any case, whatever the information retrieval system does, if a document is judged by the user to be of interest, it is relevant; otherwise, it is non-relevant. Since many factors may influence the judgement concerning relevance in a complex way, it is easy to see that designing an IR system within this frame of reference is very challenging. In designing the retrieval strategies, the IR researchers take the view that the systems should adopt methods that facilitate the ranking of documents in the order of their estimated usefulness to a user query.

It is common, in IR, to suppose that each document is indexed by a set of "content" identifiers that are variously known as keywords, index terms, subject indicators or concepts. This would require the application of some automatic or manual indexing technique to the full text or some surrogate (e.g. abstract) of the documents in order to identify the index terms to be used in their representation. In addition to the selection of index terms to represent documents, it is also common to associate weights that reflect the importance of each term as an indicator of the content of the documents to which it is assigned. The user request may, for example, be in the form of a natural language statement or a Boolean expression and in either case the query may be represented, within the IR system, as a set of (index term, weight) pairs.

The retrieval operation often consists of matching sets of index terms assigned to the stored documents with the keywords representing the user query. The matching is followed by the retrieval of those documents whose content identifiers exhibit a sufficiently high degree of similarity to the keywords of the user query. The system can obtain feedback from the user as to the appropriateness of the initial set of retrieved documents to construct improved query formulation. This kind of operation, known as relevance feedback, is particularly helpful in obtaining more effective retrieval output.

In the past, several mathematical models for document retrieval systems have been developed [C82, S83, S83a, T76, WO84]. These models are used to formally represent the basic characteristics, functional components, and the retrieval processes of document retrieval systems. Two basic categories of models that have been employed in information retrieval are the vector processing models and the Boolean retrieval models.

In the conventional vector space model (VSM), proposed by Salton [S71, S83] index terms are basic vectors in a vector space. Each document or query is represented as a linear combination of these basic term vectors. The retrieval operation consists of computing the cosine similarity function between a given query vector and the set of document vectors and then ranking documents accordingly. In this approach, the interpretation that the occurrence frequency of a term in a document represents the component of the document vector along the corresponding basic term vectors is made.

The advantages of this model are that it is simple and yet powerful. The vector operations can be performed efficiently enough to handle very large collections. Furthermore, it has been shown that the retrieval effectiveness is significantly higher compared to that of the Boolean retrieval models. However, this vector model has been incorporated into very few commercial systems.

* This research was supported in part by research grants to the first three authors from NSERC of Canada

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

In the strict Boolean retrieval systems [BU81, P84] the user query normally consists of index terms that are connected by Boolean operators *AND*, *OR* and *NOT*. The advantage of using Boolean connectives is to provide a better structure to formulate the user query. The major problem in such a system is that there is no provision for associating weights of importance to the terms which are assigned either to the documents or to the queries. In other words, the representation is binary, indicating either the presence or the absence of the various index terms. The output obtained in response to a query is not ranked in any order of presumed importance to the user. In most cases, the *AND* connectives tend to be too restrictive [BU81]. Most commercially available retrieval systems essentially conform to this model.

One of the challenges for researchers in information retrieval has been to achieve greater acceptance of the vector processing models in commercial systems. The main difficulty in this connection is due to the inability of the vector processing systems to handle Boolean queries. In recent years some progress has been made in expressing Boolean queries as vectors [S83a, S83b]. If attractive ways to achieve this are advanced, it would than be possible to modify existing systems to use vector processing techniques without a great deal of cost and effort.

Another problem in the conventional vector space model is that it assumes that term vectors are orthogonal. It is generally agreed that terms are correlated and it is necessary to generalize the model to incorporate term correlations. A vector processing model termed the GVSM [WO84a, WO85] was proposed in response to this need. In the GVSM, the queries are assumed to be presented as a list of terms and corresponding weights. Thus, no provision is made for processing Boolean queries. However, the premises of the model naturally lead to a scheme for handling Boolean queries. In this paper we present the details of this scheme. This result will help achieve the aim of integrating vector processing capabilities into existing systems which use Boolean retrieval models.

The paper is organized in the following manner. In Section 2, we review the main characteristics of the GVSM. In Section 3, its connection to the strict Boolean retrieval model is explained and the characterization of strict Boolean in vector model environment is presented. This vector model for strict Boolean is then generalized to handle weighted queries and documents. In Section 4, the various ideas and models presented in the earlier sections are summarized. Then, in Section 5, the proposed scheme is compared with the p-norm model [S83a, S83b]. This section also presents experimental results which show that the proposed scheme is effective. The final section offers some concluding remarks and areas for further research.

2. REVIEW OF THE GVSM

2.1. Vector Representation of Elements of a Boolean Algebra

Let $x_1, x_2, \ldots, x_n$ be n literals used to generate the free Boolean algebra, denoted B_{2^n}. Any Boolean expression composed of these literals (using operators *AND*, *OR* or *NOT*) is an element of the algebra.

What we desire is to identify a vector space such that every Boolean expression in B_{2^n} corresponds to a vector in the vector space. In a vector space it is necessary to specify a set of vectors that form a basis. Clearly, if a basis is known then any vector in the space can be expressed as a linear combination of the basic vectors. Since the intent is to obtain a way of expressing every possible Boolean expression, it is appropriate to have the set of basic vectors correspond to a set of fundamental expressions which can be combined to generate any element of the algebra. We, therefore, employ the notion of an atomic expression.

An atomic expression, or a minterm, in the n literals $x_1, x_2, \ldots, x_n$ is a conjunction of the literals where each x_i appears exactly once and is either in complemented or uncomplemented form. Clearly, there are 2^n minterms in all. It is well known that the conjunction of any two minterms is always zero (*false*) and that any Boolean expression in the literals $x_1, x_2, \ldots, x_n$ can be uniquely expressed as a disjunction of minterms. The representation obtained in this way is the well known disjunctive normal form [GE82].

Let $\{m_k\}_{2^n}$ denote the set of minterms in B_{2^n}. In order to characterize a vector space in which these correspond to the basic vectors, we define a set of 2^n-dimensional vectors $\{\vec{m}_k\}$. These vectors constitute an orthonormal basis of the vector space in R^{2^n} as follows:

$$\begin{aligned} \vec{m}_1 &= (1,0,0,\ldots,0) \\ \vec{m}_2 &= (0,1,0,\ldots,0) \\ \vec{m}_3 &= (0,0,1,\ldots,0) \\ &\vdots \\ \vec{m}_{2^n} &= (0,0,0,\ldots,1) \end{aligned} \qquad (2.1)$$

Given these, it is easily seen that the vector representation of any Boolean expression is given by the vector sum of the basic vectors which correspond to the minterms in the disjunctive normal form of the expression.

The assertion that, for any two vectors $\vec{m}_i$, $\vec{m}_j$, the scalar product $\vec{m}_i \cdot \vec{m}_j$ is *false* corresponds to the fact that the conjunction of atomic expressions m_i and m_j is *false*. In general, if two vectors are not orthogonal, then the corresponding Boolean expressions have at least one minterm in common.

2.2. Vector Representation of Terms Assuming No Weights

The ideas developed in Section 2.1 can be applied to an information retrieval environment and each index term can be given an explicit vector representation. Let the indexing vocabulary consist of terms(literals) $t_1, t_2, \ldots, t_n$. Any literal can appear in a Boolean expression either as $\bar{t}_i$ or t_i, depending on whether it needs to be complemented or not. In particular, conjunctive expressions where every literal appears in either uncomplemented or complemented form are the atomic expressions.

Let $\{m_k\}_{2^n}$ denote the set of all atomic expressions. Then, since each t_i is itself an element of the Boolean algebra generated, t_i can be expressed in its disjunctive normal form:

$$t_i = m_{i_1} \; OR \; m_{i_2} \ldots OR \; m_{i_r} \; , \qquad (2.2)$$

where the m_{i_j}'s are minterms in which t_i is uncomplemented. Let the set of minterms in eqn. (2.2) be denoted by $\{m\}^i$. We can now define the basic vectors analogous to eqn. (2.1) and the term t_i can be written in the vector notation as

$$\vec{t}_i = \sum_{m_k \in \{m\}^i} \vec{m}_k \; . \qquad (2.3)$$

Alternatively,

$$\vec{t}_i = \sum_{i=1}^{2^n} c_{i_k} \vec{m}_k \qquad (2.4)$$

where

$$c_{i_k} = \begin{cases} 1 & \text{if } m_k \in \{m\}^i \\ 0 & otherwise \; . \end{cases}$$

That is, the term vectors are a linear combination of the $\vec{m}_k$'s, the basic vectors, and the vector summation corresponds to the *OR*'s of eqn. (2.2). Furthermore, the scalar product between any two basic vectors is zero corresponding to the fact that the *AND*ing of two minterms is *false*.

2.3. The Generalized Vector Space Model (GVSM)

In this section we will review the essential features of the GVSM [WO84, WO85]. This model is the result of incorporating the idea developed in Section 2.2 into the framework of the conventional vector space model. One of the main steps in this process involves the generalization of the term vector representation in such a way that the expansion coefficients in eqn. (2.4) are not binary. The determination of these coefficients is, however, closely tied in with the question of what is meant by two terms being not orthogonal (or, correlated). This is because, once the coefficients are specified, the scalar product between any two non-binary vectors $\vec{t}_i$ and $\vec{t}_j$ is defined. Since scalar product being zero implies orthogonality, a non-zero value must represent a measure of non-orthogonality. In order to motivate the premises of the GVSM and to introduce some basic concepts, we first outline the main ideas of the conventional vector space model.

2.3.1. The conventional vector space model

The basic premise in the vector space model is that the documents and the queries are represented by a set of vectors, say, $\{ \vec{d}_\alpha \mid \alpha = 1, 2, \ldots, p\}$, and $\vec{q}$, respectively, in a vector space spanned by the set of normalized term vectors, $\{ \vec{t}_i \mid i = 1, 2, \ldots, n \}$. That is,

$$\vec{d}_\alpha = \sum_{i=1}^{n} a_{\alpha i} \vec{t}_i , \qquad (\alpha = 1, 2, \ldots, p) , \tag{2.5}$$

$$\vec{q} = \sum_{j=1}^{n} q_j \vec{t}_j . \tag{2.6}$$

Given the above representation, the scalar product $\vec{d}_\alpha \cdot \vec{q}$ (this becomes the cosine of the angle between $\vec{d}_\alpha$ and $\vec{q}$ if the vectors are normalized), which may serve as a measure of the similarity between each document in $\{\vec{d}_\alpha\}_p$ and the query $\vec{q}$, is defined by

$$\vec{d}_\alpha \cdot \vec{q} = \sum_{i=1, j=1}^{n} a_{\alpha i} q_j \vec{t}_i \cdot \vec{t}_j , \qquad (\alpha = 1, 2, \ldots, p) . \tag{2.7}$$

The higher the similarity value of a document to a query, the greater is its potential relevance to the user query. The retrieval strategy consists of obtaining ranked output of documents in the decreasing order of the query-document similarities such that the documents with higher similarity values will be retrieved first. However, for this purpose it is necessary to know both the correlation between any pair of vectors in the basis, $\{ \vec{t}_i \}_n$ and the components of documents and queries along these basic vectors.

It is convenient for subsequent discussions to express eqn. (2.5) in matrix notation as follows :

$$D = t A^T , \tag{2.8}$$

where

A^T is the transpose of A, $D = (\vec{d}_1, \vec{d}_2, \ldots, \vec{d}_p)$, $t = (\vec{t}_1, \vec{t}_2, \ldots, \vec{t}_n)$, and

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ a_{p1} & a_{p2} & \cdots & a_{pn} \end{bmatrix} \tag{2.9}$$

Similarly, eqn. (2.7) can be rewritten as

$$S = q\, G\, A^T , \tag{2.10}$$

where $S = (\vec{d}_1 \cdot \vec{q}, \vec{d}_2 \cdot \vec{q}, \ldots, \vec{d}_p \cdot \vec{q})$, $q = (q_1, q_2, \ldots, q_n)$, and

$$G = (g_{ij}) = \begin{bmatrix} \vec{t}_1 \cdot \vec{t}_1 & \vec{t}_1 \cdot \vec{t}_2 & \cdots & \vec{t}_1 \cdot \vec{t}_n \\ \vec{t}_2 \cdot \vec{t}_1 & \vec{t}_2 \cdot \vec{t}_2 & \cdots & \vec{t}_2 \cdot \vec{t}_n \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \vec{t}_n \cdot \vec{t}_1 & \vec{t}_n \cdot \vec{t}_2 & \cdots & \vec{t}_n \cdot \vec{t}_n \end{bmatrix} \tag{2.11}$$

In the conventional vector space model, the term occurrence frequency matrix obtained empirically from automatic indexing is assumed to be the matrix A. Since the correlations between terms are not known a priori, as a first order of approximation, the term vectors are assumed to be orthogonal to each other. Thus the correlation matrix G, defined in eqn. (2.11), becomes an identity matrix. That is,

$$g_{ij} = \vec{t}_i \cdot \vec{t}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j . \end{cases}$$

With such approximations (i.e. $G = I$), the *ranking matrix* S for a given query $\vec{q}$ with respect to a set of document vectors can, therefore, be computed easily from the following equation :

$$S = q\, A^T . \tag{2.12}$$

2.3.2. Document and query representation in GVSM

The discussion on VSM is useful in showing that the document representation in GVSM will, under proper restriction, reduce to that of VSM. The fact that we want terms to have an explicit representation renders the computation of the scalar product between term vectors direct. Thus, GVSM provides a prescription for computing term correlations.

From eqn. (2.5), it is seen that in VSM the representation of a document is taken to be a vector sum of term vectors. In GVSM, we introduce the "circle" sum operator (denoted by $\oplus$), and hypothesize that a document should be expressed as the circle sum of the associated term vectors. More precisely,

$$\vec{d}_\alpha = a_{\alpha 1}\vec{t}_1 \oplus a_{\alpha 2}\vec{t}_2 \oplus \ldots a_{\alpha n}\vec{t}_n$$

$$= \bigoplus_{i=1}^{n} a_{\alpha i} \vec{t}_i , \tag{2.13}$$

where the circle sum $\oplus$ is defined as follows. Let $\vec{t}_r$ and $\vec{t}_s$ be vectors having the form specified in eqn (2.4). Then,

$$\vec{t}_r \oplus \vec{t}_s = \sum_{m_r, m_s \in K} max(c_{rm_r}, c_{sm_s})\, \vec{m}_r , \tag{2.14}$$

where $K = \{m\}^r \cup \{m\}^s$. This choice for the representation of $\vec{d}$'s is intended to reflect the fact that terms are correlated. Let us consider, for example, the situation where terms are not correlated. That is,

$$\vec{t}_r \cdot \vec{t}_s = 0 ,$$

implying that there are no common basic vectors in the expansions of

$\vec{i}_r$ and $\vec{i}_s$, $1 \le r, s \le n$. Clearly, in this case the document representation immediately reduces to the vector sum of $\vec{i}$'s as defined by eqn. (2.5). Thus, the representation proposed in eqn. (2.13) is, in essence, a generalization of the premise that the document vectors may be expressed as a vector sum of the $\vec{m}$'s. It is proposed that the coefficients of the $\vec{m}$'s should be obtained by first prescribing an expansion for each term vector and then by using eqn. (2.13).

It is also necessary to specify the way in which a query will be represented. Given $q = (q_1, q_2, \ldots, q_n)$, we propose that the query be represented as the vector sum of the $\vec{i}_j$'s involved. That is,

$$\vec{q} = \sum_{j=1}^{n} q_j \vec{i}_j . \qquad (2.15)$$

This choice is made since, in the VSM, the user does not imply any structure in the query specification. As we shall see later, when it is necessary to incorporate more structure, queries will not be given by the above expression.

Using these prescriptions, both documents and queries can be expressed as a linear combination of the $\vec{m}$'s and the computation of $\vec{d}_m \cdot \vec{q}$ is straightforward. All that still remains is to show how eqn. (2.4) is generalized to express the $\vec{i}$'s as a vector sum of $\vec{m}$'s. As mentioned earlier, this requires the meaning of term correlations to be made precise.

2.3.3. Computation of term correlations

First a simple example is presented to motivate the approach adopted.

Example 2.1 Let D be a set of documents indexed only by two terms, t_1 and t_2. Let D_F be the maximal subset of documents satisfying F, where F is a Boolean expression in the t's.

We can identify the following disjoint subsets forming a partition of D:

$$D_{t_1\bar{t}_2} = D_{t_1} \cap \overline{D}_{t_2} ,$$

$$D_{t_1t_2} = D_{t_1} \cap D_{t_2} ,$$

$$D_{\bar{t}_1t_2} = \overline{D}_{t_1} \cap D_{t_2} ,$$

where $D_{t_1\bar{t}_2}$, $D_{t_1t_2}$, etc. correspond respectively to $D_{t_1 \text{ AND } \bar{t}_2}$, $D_{t_1 \text{ AND } t_2}$, etc. The *AND*'s are dropped for convenience. $\overline{D}_{t_i}$ denotes the set complement of D_{t_i} (i.e. $\overline{D}_{t_i}$ is the subset of documents not containing t_i) .

Based on intuition we argue that the correlation between any two index terms depends on the number of documents in which these two terms appear together. This sort of argument based on term co-occurrence information has been the basis for measuring term correlation in earlier studies [V77]. Let $c(D_F)$ denotes the cardinality of the set D_F. The cardinality $c(D_{t_1t_2})$, of the subset $D_{t_1t_2} = D_{t_1} \cap D_{t_2}$ (which denotes the number of documents containing t_1 and t_2) thus provides a plausible measure of the "unnormalized" correlation between t_1 and t_2 .

In terms of vector notation, the correlation between t_1 and t_2 , denoted by $\vec{t}_1 \cdot \vec{t}_2$, can be conveniently expressed as the scalar product of two normalized term vectors, $\vec{t}_1$ and $\vec{t}_2$, namely ,

$$\vec{t}_1 \cdot \vec{t}_2 = \frac{c^2(D_{t_1t_2})}{[c^2(D_{t_1\bar{t}_2}) + c^2(D_{t_1t_2})]^{1/2} \; [c^2(D_{\bar{t}_1t_2}) + c^2(D_{t_1t_2})]^{1/2}} ,$$

where

$$\vec{t}_1 = \frac{c(D_{t_1\bar{t}_2})\, \vec{m}_1 + c(D_{t_1t_2})\, \vec{m}_2}{[c^2(D_{t_1\bar{t}_2}) + c^2(D_{t_1t_2})]^{1/2}} ,$$

$$\vec{t}_2 = \frac{c(D_{t_1t_2})\, \vec{m}_2 + c(D_{\bar{t}_1t_2})\, \vec{m}_3}{[c^2(D_{\bar{t}_1t_2}) + c^2(D_{t_1t_2})]^{1/2}} ,$$

and $\vec{m}_1$, $\vec{m}_2$, and $\vec{m}_3$ are the orthonormal basic vectors . ■

It is evident from the example that terms can be meaningfully expressed as linear combinations of $\vec{m}$'s. Clearly, $\vec{m}_1$, $\vec{m}_2$ and $\vec{m}_3$ correspond respectively to the atomic expressions $t_1\,\bar{t}_2$, $t_1\,t_2$ and $\bar{t}_1\,t_2$. In the example, only the presence or absence of a term in a document is considered. This limitation is reflected in the assertion that $c(D_{t_1t_2})$ is a measure of the correlation between $\vec{t}_1$ and $\vec{t}_2$. Furthermore, the example helps in convincing oneself that the expansion of a term vector, say $\vec{t}_i$, need not have a non-zero coefficient for all vectors corresponding to the minterms in $\{m\}^i$. This is due to the fact that term co-occurrences depend on the particular collection of documents at hand. For instance, if t_1 and t_2 do not co-occur in a given collection then the expansions of neither $\vec{t}_1$ nor $\vec{t}_2$ will involve $\vec{m}_2$, and $\vec{t}_1 \cdot \vec{t}_2$ will be zero.

More generally, given terms $t_1, t_2, \ldots, t_n$ and a collection of documents of cardinality p, the number of "active" minterms is a subset of all possible minterms, $\{m\}_{2^n}$. Since, in the worst case, each document can correspond to a different minterm, the number of active basic vectors is at most p. Thus, the expansion of $\vec{i}_i$, $1 \le i \le n$, involves only those basic (atomic) vectors restricted to the set of active minterms. It is understood, in subsequent discussions, that $\{\vec{m}\}^i$ refers to this subset of active basic vectors corresponding to $\vec{i}_i$.

Another issue raised in the discussion above is the limitation of using only the cardinalities. A natural generalization, which considers the importance of a term to the documents (i.e. term weights), should be developed. In [WO85], the following expression for (unnormalized) $\vec{i}_i$ is proposed:

$$\vec{i}_i = \sum_{m_k \in \{m\}^i} c_{i_k}\, \vec{m}_k \qquad (2.16)$$

where the unnormalized form of c_{i_k} is given by $\sum_{d_m \in D_{m_k}} a_{dm}$. The above concepts are illustrated with an example.

Example 2.2 Given a set of documents $D = \{ d_1, d_2, d_3, d_4 \}$ indexed by the set of terms $T = \{ t_1, t_2, t_3 \}$. The weights of each term in the documents are given by the following matrix:

$$A = \begin{bmatrix} & t_1 & t_2 & t_3 \\ d_1 & 2 & 0 & 1 \\ d_2 & 1 & 0 & 0 \\ d_3 & 0 & 1 & 3 \\ d_4 & 2 & 0 & 0 \end{bmatrix}$$

Without loss of generality, the 8 fundamental products or minterms can be represented as follows:

$$m_1 = t_1\,\bar{t}_2\,t_3\,, \qquad m_2 = t_1\,\bar{t}_2\,\bar{t}_3\,,$$

$$m_3 = \bar{t}_1\,t_2\,t_3\,, \qquad m_4 = t_1\,t_2\,t_3\,,$$

$$m_5 = t_1\,t_2\,\bar{t}_3\,, \qquad m_6 = \bar{t}_1\,\bar{t}_2\,t_3\,,$$

$$m_7 = \bar{t}_2\,t_2\,\bar{t}_3\,, \qquad m_8 = \bar{t}_1\,\bar{t}_2\,\bar{t}_3\,.$$

Each $t_i \in T$ can be expressed in a disjunctive normal form:

$$t_1 = t_1 \text{ AND } (t_2 \text{ OR } \bar{t}_2) \text{ AND } (t_3 \text{ OR } \bar{t}_3)$$

$$= [(t_1 \text{ AND } t_2) \text{ OR } (t_1 \text{ AND } \bar{t}_2)]$$

$$\text{AND } (t_3 \text{ OR } \bar{t}_3)$$

$$= (t_1 \text{ AND } t_2 \text{ AND } t_3) \text{ OR}$$

$$(t_1 \text{ AND } \bar{t}_2 \text{ AND } t_3) \text{ OR}$$

$$(t_1 \text{ AND } t_2 \text{ AND } \bar{t}_3) \text{ OR}$$

$$(t_1 \text{ AND } \bar{t}_2 \text{ AND } \bar{t}_3)$$

$$= m_4 \text{ OR } m_1 \text{ OR } m_5 \text{ OR } m_2\,,$$

$$t_2 = t_2 \text{ AND } (t_1 \text{ OR } \bar{t}_1) \text{ AND } (t_3 \text{ OR } \bar{t}_3)$$

$$= m_4 \text{ OR } m_3 \text{ OR } m_5 \text{ OR } m_7\,,$$

and

$$t_3 = t_3 \text{ AND } (t_1 \text{ OR } \bar{t}_1) \text{ AND } (t_2 \text{ OR } \bar{t}_2)$$

$$= m_4 \text{ OR } m_1 \text{ OR } m_3 \text{ OR } m_6\,.$$

From eqn. (2.16) and the above document matrix A, we obtain

$$\vec{t}_1 = \frac{2\vec{m}_1 + (1+2)\vec{m}_2 + 0\vec{m}_3}{\sqrt{2^2 + 3^2}}$$

$$= 0.55\vec{m}_1 + 0.83\vec{m}_2$$

$$\vec{t}_2 = \frac{0\vec{m}_1 + (0+0)\vec{m}_2 + 1\vec{m}_3}{\sqrt{1^2}}$$

$$= \vec{m}_3$$

$$\vec{t}_3 = \frac{1\vec{m}_1 + (0+0)\vec{m}_2 + 3\vec{m}_3}{\sqrt{1^2 + 3^2}}$$

$$= 0.32\vec{m}_1 + 0.95\vec{m}_3$$

By substituting the above expressions for term vectors into the eqn. (2.13), it follows:

$$\vec{d}_1 = 2\vec{t}_1 \oplus \vec{t}_3$$

$$= 2(0.55\vec{m}_1 + 0.83\vec{m}_2) \oplus (0.32\vec{m}_1 + 0.95\vec{m}_3)$$

$$= 1.10\vec{m}_1 + 1.66\vec{m}_2 + 0.95\vec{m}_3$$

$$\vec{d}_2 = \vec{t}_1 = 0.55\vec{m}_1 + 0.83\vec{m}_2$$

$$\vec{d}_3 = \vec{t}_2 \oplus 3\vec{t}_3 = \vec{m}_3 \oplus 3(0.32\vec{m}_1 + 0.95\vec{m}_3)$$

$$= 0.96\vec{m}_1 + 2.85\vec{m}_3$$

$$\vec{d}_4 = 2\vec{t}_1 = 2(0.55\vec{m}_1 + 0.83\vec{m}_2)$$

$$= 1.1\vec{m}_1 + 1.66\vec{m}_2\,.$$

Similarly, we can transform, for example, the query vector, $\vec{q} = \vec{t}_1 + \vec{t}_2$, into a linear combination of atomic vectors :

$$\vec{q} = (0.55\vec{m}_1 + 0.83\vec{m}_2) + (\vec{m}_3)$$

$$= 0.55\vec{m}_1 + 0.83\vec{m}_2 + \vec{m}_3\,.$$

Then, the cosine similarity $\dfrac{\vec{d}_i \cdot \vec{q}}{\|\vec{d}_i\|\,\|\vec{q}\|}$ between the document d_α and the query q, can be computed as follows:

$$s_1' = \frac{(1.10)(.55) + (1.66)(.83) + (.95)(1)}{\sqrt{1.10^2 + 1.66^2 + 0.95^2}\sqrt{0.55^2 + 0.83^2 + 1^2}}$$

$$= 0.9248\,.$$

$$s_2' = \frac{(.55)(.55) + (.83)(.83) + (0)(1)}{\sqrt{0.55^2 + 0.83^2}\sqrt{0.55^2 + 0.83^2 + 1^2}}$$

$$= 0.7056\,.$$

$$s_3' = \frac{(.96)(.55) + (0)(.83) + (2.85)(1)}{\sqrt{0.96^2 + 3.85^2}\sqrt{0.55^2 + 0.83^2 + 1^2}}$$

$$= 0.7819\,.$$

$$s_4' = \frac{(1.1)(.55) + (1.66)(.83) + (0)(1)}{\sqrt{1.1^2 + 1.66^2}\sqrt{0.55^2 + 0.83^2 + 1^2}}$$

$$= 0.7056\,.$$

Based on these cosine similarity values ($s_1' > s_3' > s_2' \geq s_4'$), d_1 will be retrieved first, d_3 second and so on. ■

3. PROCESSING BOOLEAN QUERIES IN THE GVSM

In GVSM, a query is simply defined as a weighted vector sum of term vectors. Unfortunately, the choice of this form of query does not facilitate the user to explicitly specify structure as can be done in Boolean systems. Thus, while the most common language used in commercial systems to express query structure involves Boolean logic, in GVSM the query is considered to be a list of (index term, weight) pairs. Clearly, it would be advantageous to provide a prescription to handle Boolean queries in the GVSM environment. In this section we develop such a prescription. In order to demonstrate that the proposed scheme is natural and sound, we first characterize the strict Boolean retrieval system within the framework of the generalized vector space model.

3.1. Strict Boolean Retrieval in the GVSM Environment

A standard approach to describing Boolean retrieval environment is to characterize any document as a set of terms and queries as Boolean expressions on the variables associated with the terms.

Now, if a document d_α is defined as the set of terms, $\{t_{\alpha 1}, t_{\alpha 2}, \ldots, t_{\alpha k}\}$, then d_α is said to satisfy a given query provided that the Boolean expression of the query evaluates to 1 by substituting a value 1 only for the variables $t_{\alpha 1}, t_{\alpha 2}, \ldots, t_{\alpha k}$ and a value 0 for the remaining ones. An alternative way to interpret this retrieval strategy is to regard each document as being represented by an atomic expression. That is, each document is represented by the minterm in which exactly those terms contained by the document are unnegated. In subsequent

discussions, the minterm which is associated with a document in this manner is referred to as the *dominant* atom of the document. Then, a document is said to satisfy a query only if the atomic expression representing the document appears as one of those atoms in the disjunctive normal form of the query expression.

In terms of the above characterization of documents, the mapping of the strict Boolean retrieval environment to the vector space environment is straightforward. Specifically, each document maps to the basic vector $\vec{m}_k$ corresponding to the document's dominant atom. On the other hand, a query (a Boolean expression) can be transformed into vector sum of the basis vectors corresponding to those atoms that appear in its disjunctive normal form. Thus, in this framework, only those document vectors whose scalar product with a given query vector is non-zero would be retrieved.

Note that the representation of a document by a single atomic vector is a special case of that adopted in GVSM. In general, a document is represented by a vector sum of atomic vectors:

$$\vec{d}_\alpha = \sum_{K=1}^{2^n} c_{\alpha_k} \vec{m}_k .$$

For the strict Boolean environment, all the c_{α_k}'s but one are zero and d_α is assumed to be modelled by its dominant atom. Clearly, this discussion highlights the point that, within our framework, the document vectors are treated as being orthogonal to each other in the strict Boolean environment. In other words, hypothesizing documents as a vector sum of $\vec{m}$'s is a generalization which recognizes that documents may not necessarily be represented by orthogonal vectors.

3.2. Extended Boolean Environment under the GVSM

In this section we extend the ideas presented to obtain a scheme for expressing weighted Boolean queries as a vector. As far as the documents are concerned, the representation is the same as that for the GVSM. That is,

$$\vec{d}_\alpha = \bigoplus_{i=1}^{n} a_{\alpha i} \vec{i}_i .$$

Before we describe the transformation of weighted Boolean queries, a query language consisting of weighted terms linked by *AND*, *OR*, and *NOT* connectives is introduced.

From a set of index terms, $\{t_i \mid i = 1, 2, \ldots, n\}$, a simple query is defined as an ordered pair

$$q_i = (t_i, w_i) , \qquad (0 \le w_i \le 1),$$

where w_i denotes the term weight. A general (mixed) query can be formed from a set of simple queries, $\{q_1, q_2, \ldots\}$, linked by the *AND*, *OR* and *NOT* connectives. For example, a complex query q can be written as

$$q = [NOT\ q_1\ AND\ (q_2\ OR\ q_3)]\ OR\ q_4$$

$$= [NOT(t_1, w_1)\ AND\ (t_2, w_2)\ OR\ (t_3, w_3)]\ OR\ (t_4, w_4) .$$

The next step is to define a suitable mapping which transforms such query q to a vector in R^{2^n}.

First, it is necessary to define the following operators $\oplus$, $\odot$ and $\neg$ for disjunction,conjunction and complementation in a vector subspace R^{2^n}. Let

$$M = \bigcup_{i=1}^{n} \{m\}^i ,$$

where $\{m\}^i$ denotes the set of active minterms, which spans t_i. That is, M is the set of basic vectors which span all the term vectors in $\{\vec{i}\}_n$. In the previous section, to generalize a vector processing system, it has been shown that each term vector can be expressed as a linear combination of its minterms or basic vectors in the vector subspace R^{2^n}. That is,

$$\vec{i}_i = \sum_{m_k \in \{m\}^i} c_{i_k} \vec{m}_k ,$$

where the c_{i_k}'s are normalized coefficients such that $0 \le c_{i_k} \le 1$.

(i) For the unary operator $\neg$ on a vector $\vec{i}_i$ spanned by M is defined as follows

$$\begin{aligned} \neg \vec{i}_i &= I - \vec{i}_i \\ &= I - \sum_{m_k \in \{m\}^i} c_{i_k} \vec{m}_k \\ &= \sum_{m_k \in \{m\}^i} (1 - c_{i_k}) \vec{m}_k + \sum_{m_k \in M - \{m\}^i} \vec{m}_k , \end{aligned} \tag{3.1}$$

where

$$I = \sum_{m_k \in M} \vec{m}_k .$$

(ii) Given any two term vectors $\vec{i}_1$, $\vec{i}_2$:

$$\vec{i}_1 = \sum_{m_k \in \{m\}^1} c_{1_k} \vec{m}_k ,$$

$$\vec{i}_2 = \sum_{m_k \in \{m\}^2} c_{2_k} \vec{m}_k ,$$

the disjunction operator $\oplus$ is defined by

$$\vec{i}_1 \oplus \vec{i}_2 = \sum_{m_k \in \{m\}^1 \cup \{m\}^2} max(c_{1_k}, c_{2_k})\, \vec{m}_k . \tag{3.2}$$

(iii) Similarly, the conjunction operator $\odot$ is defined by

$$\vec{i}_1 \odot \vec{i}_2 = \sum_{m_k \in \{m\}^1 \cap \{m\}^2} max(c_{1_k}, c_{2_k})\, \vec{m}_k . \tag{3.3}$$

The above definitions of $\oplus$ and $\neg$ are straightforward. In earlier work, the operator analogous to the circle-product operator, $\odot$, has been defined by *min*. We instead propose *max*, but restricted to those basic vectors common to both $\vec{i}_1$ and $\vec{i}_2$. Researchers working on the fuzzy set model have argued that using the *min* function unduly subdues the importance of the common elements [WA79, S83a]. We propose that the use of *max* function would alleviate this problem.

Having defined the operations $\neg$, $\oplus$ and $\odot$, it is necessary to introduce a mapping, h: $\{q\} \rightarrow R^{2^n}$ (which is used to transform weighted Boolean query expressions into vectors) as follows:

$$\begin{aligned} &(a)\ h(t_i, w_i) &&= w_i \vec{i}_i \\ &(b)\ h[NOT\ (q)] &&= \neg\, h(q) \\ &(c)\ h[q_1\ OR\ q_2] &&= h(q_1) \oplus h(q_2) \\ &(d)\ h[q_1\ AND\ q_2] &&= h(q_1) \odot h(q_2) \end{aligned} \tag{3.4}$$

Based on the mapping h defined above, the GVSM query language satisfies the algebraic properties of idempotency, commutativity, duality, involution, associativity and distributivity (with one exception). The proofs of these properties can be found in [WO84].

In the GVSM Boolean system, the complement laws that hold for the strict Boolean expressions are not valid, that is

$$h[(t_i, w_i)\ AND\ [NOT\ (t_i, w_i)]] \neq \emptyset ,$$

$$h[(t_i, w_i)\ OR\ [NOT\ (t_i, w_i)]] \neq I .$$

However, these two properties are not essential for the retrieval operations. In addition, the GVSM approach for Boolean system can also be applied to a situation where different weights of importance must be assigned to phrases or query clauses. For example, sp

$$[(t_1, w_1)\ AND\ ((t_2, w_2), w_\alpha)]\ OR\ [(t_3, w_3)\ AND\ ((t_4, w_4), w_\beta)]\ ,$$

where w_α, w_β are the weights for the phrases with terms $[t_1, t_2]$ and $[t_3, t_4]$ respectively. This property is particularly important to provide an automatic environment where a Boolean expression can be formulated automatically from the initial available natural language statements [S84].

The transformation of a Boolean expression into a vector query can be illustrated with an example.

Example 3.1 Let $M = \{\vec{m}_1, \vec{m}_2, \vec{m}_3, \vec{m}_4\}$ be a set of atomic vectors which span the following term vectors:

$$\vec{t_1} = 0.4\vec{m}_1 + 0.2\vec{m}_2 + 0.1\vec{m}_4\ ,$$
$$\vec{t_2} = 0.2\vec{m}_2 + 0.3\vec{m}_3\ ,$$
$$\vec{t_3} = 0.1\vec{m}_3 + 0.3\vec{m}_4\ .$$

Based on our formalism, the results of the transformation of Boolean queries into vector queries are summarized in Table 3.1. Some of the entries of the table are explained in detail below. Consider, for example, a simple Boolean query, $(t_3,0.3)$. By the mapping h defined in eqn. (3.4), such a query can be transformed into the following vector:

$$h[(t_3,0.3)] = 0.3\vec{t_3}$$
$$= (0.3)(0.1\vec{m}_3 + 0.3\vec{m}_4)$$
$$= 0.03\vec{m}_3 + 0.09\vec{m}_4\ .$$

Applying eqn. (3.1) for the *NOT* operator, we obtain

$$h[NOT(t_3,0.3)] = (1-0)\vec{m}_1 + (1-0)\vec{m}_2 + (1-0.03)\vec{m}_3 + (1-0.09)\vec{m}_4$$
$$= \vec{m}_1 + \vec{m}_2 + 0.97\vec{m}_3 + 0.91\vec{m}_4\ .$$

A more complex query, for example, $(t_1,0.2)AND(t_3,0.3)$, can be transformed into the vector:

$$h[(t_1,0.2)AND(t_3,0.3)]$$
$$= (0.08\vec{m}_1 + 0.04\vec{m}_2 + 0.02\vec{m}_4) \odot (0.03\vec{m}_3 + 0.09\vec{m}_4)$$
$$= max(0.02, 0.09)\vec{m}_4 = 0.09\vec{m}_4\ .$$

Similarly, eqn. (3.2) can be used to evaluate an *OR* query, $(t_2,0.05)OR(t_3,0.3)$. We obtain

$$h[(t_2,0.5)OR(t_3,0.3)]$$
$$= (0.1\vec{m}_2 + 0.15\vec{m}_3) \oplus (0.03\vec{m}_3 + 0.09\vec{m}_4)$$
$$= 0.1\vec{m}_2 + max(0.15,0.03)\vec{m}_3 + 0.09\vec{m}_4$$
$$= 0.1\vec{m}_2 + 0.15\vec{m}_3 + 0.09\vec{m}_4\ .$$

It can also be shown that the distributive rule of the *AND* over *OR* holds. For example,

$$h[(t_1,0.2)\ AND\ [(t_2,0.5)\ OR\ (t_3,0.3)]] =$$
$$h[[(t_1,0.2)\ AND\ (t_2,0.5)]$$
$$OR\ [(t_1,0.2)\ AND\ (t_3,0.3)]]\ .$$

However the distributive rule does not hold for *OR* over *AND* as shown below:

$$h[(t_1,0.2)\ OR\ [(t_2,0.5)\ AND\ (t_3,0.3)]] \neq$$
$$h[[(t_1,0.2)\ OR\ (t_2,0.5)]$$
$$AND\ [(t_1,0.2)\ OR\ (t_3,0.3)]]\ .\ \blacksquare$$

Thus, using the method mentioned in Example 2.2 of the previous section, any query vector obtained from the above transformation can be used to compute the query-document similarity values.

Boolean queries	vector queries			
expressions	$\vec{m}_1$	$\vec{m}_2$	$\vec{m}_3$	$\vec{m}_4$
(t_1)	0.4	0.2		0.1
(t_2)		0.2	0.3	
(t_3)			0.1	0.3
$(t_1, 0.2)$	0.08	0.04		0.02
$(t_2, 0.5)$		0.1	0.15	
$(t_3, 0.3)$			0.03	0.09
NOT$(t_3, 0.3)$	1.0	1.0	0.97	0.91
$(t_1, 0.2)$AND$(t_2, 0.5)$		0.1		
$(t_1, 0.2)$AND$(t_3, 0.3)$				0.09
$(t_2, 0.5)$OR$(t_3, 0.3)$		0.1	0.15	0.09
$(t_2, 0.5)$AND$(t_3, 0.3)$			0.15	
$([(t_2, 0.5)$OR$(t_3, 0.3)], 0.8)$		0.08	0.12	0.072
$(t_1,0.2)$AND$[(t_2,0.5)$OR$(t_3,0.3)]$		0.1		0.09
$[(t_1,0.2)$AND$(t_2,0.5)]$OR$[(t_1,0.2)$AND$(t_3,0.3)]$		0.1		0.09
$(t_1,0.2)$OR$[(t_2,0.5)$AND$(t_3,0.3)]$	0.08	0.04	0.15	0.02
$[(t_1,0.2)$OR$(t_2,0.5)]$AND$[(t_1,0.2)$OR$(t_3,0.3)]$		0.1	0.15	0.09

Table 3.1. Examples of Boolean query transformation.

4. AN OVERVIEW OF GVSM AND ITS VARIATIONS

The ideas developed in the last two sections are summarized. The basic premises of the model are characterized in how terms and documents are represented. That is,

$$\vec{t}_i = \sum_{m_k \in (m)^i} c_{i_k} \vec{m}_k$$

$$\vec{d}_\alpha = \bigoplus_{i=1}^{n} a_{\alpha i} \vec{t}_i .$$

Two important variations are possible depending on whether the users need to have the option of structuring queries. The two cases are :

(i) Queries are specified as a list of (index term, weight) pairs. In this case we have,

$$\vec{q} = \sum_{j=1}^{n} q_j \vec{t}_j .$$

(ii) Queries are specified as a weighted Boolean expression in which (index term, weights) are join by the connectives *AND*, *OR* and *NOT*. The query vector is then given by

$$\vec{q} = h(q) .$$

The first formulation is the *basic* GVSM and the second one characterises the *unified* GVSM. The latter makes it possible to process extended Boolean queries within our framework.

An important aspect of the basic GVSM is that it generalizes VSM to incorporate term correlations. In fact, we have pointed out that, when terms are assumed to be orthogonal, document representation reduces to the vector sum of terms. Similarly, the unified GVSM reduces to the strict Boolean retrieval model when each document is represented by its *dominant* atomic vector. Of course, identifying each document with a single atomic vector implies that the documents are orthogonal to each other.

5. EVALUATION OF THE UNIFIED GVSM

In this section we provide an evaluation of the unified model. The evaluation is done in two stages. We, first, present a brief description of the p-norm model, which was proposed by Salton and his coworkers for extended Boolean retrieval, and contrast the models at a conceptual level. Secondly, we describe experimental results which compare the retrieval effectiveness of these models.

5.1. The p-norm Model and a Comparison of General Features

In the p-norm model [S83a, S83b], the documents are represented by a set of vectors $\{d_\alpha \mid \alpha = 1, 2, \ldots, p\}$ in a vector space spanned by a set of orthonormal term vectors $\{\vec{t}_i \mid i = 1, 2, \ldots, n\}$. Each document vector can be defined as

$$\vec{d}_\alpha = \sum_{i=1}^{n} a_{\alpha i} \vec{t}_i \qquad (5.1)$$

where $0 \le a_{\alpha i} \le 1$ denotes the weight of term i in document d_α.

The query-document similarity is measured by a generalized scalar product between corresponding vectors in the document space. This generalization uses the well-known L_p vector norm defined for an n-dimensional vector, $\vec{d}_\alpha$. The length of $\vec{d}_\alpha$ is given by

$$\|\vec{d}_\alpha\| = \|a_{\alpha 1}, a_{\alpha 2}, \ldots, a_{\alpha n}\|_p = (a_{\alpha 1}^p + a_{\alpha 2}^p + , \ldots, + a_{\alpha n}^p)^{\frac{1}{p}} ,$$

where p is an integer value ranging from 1 to ∞ .

Let w_i be the weight of query term t_i, $0 \le w_i \le 1$ and $1 \le p \le \infty$. Based on the p-norm model, the generalized Boolean *OR*-query, Q_{OR^p}, and *AND*-query, Q_{AND^p}, can be written as

$$Q_{OR^p} = [(t_1, w_1)\ OR^p\ (t_2, w_2)\ OR^p \ldots OR^p\ (t_m, w_m)] ,$$

and

$$Q_{AND^p} = [(t_1, w_1)\ AND^p\ (t_2, w_2)\ AND^p \ldots AND^p\ (t_m, w_m)] .$$

The similarity measurement between a given document $\vec{d}_\alpha = (a_{\alpha 1}, a_{\alpha 2}, \ldots, a_{\alpha n})$ and the Q_{OR^p} and Q_{AND^p} are defined respectively by:

$$sim(\vec{d}_\alpha, Q_{OR^p}) = \left[\frac{a_{\alpha 1}^p w_1^p + a_{\alpha 2}^p w_2^p + \ldots + a_{\alpha n}^p w_n^p}{w_1^p + w_2^p + \ldots + w_n^p} \right]^{\frac{1}{p}} , \qquad (5.2)$$

$$sim(\vec{d}_\alpha, Q_{AND^p}) = 1 - \left[\frac{(1-a_{\alpha 1})^p w_1^p + (1-a_{\alpha 2})^p w_2^p + \ldots + (1-a_{\alpha n})^p w_n^p}{w_1^p + w_2^p + \ldots + w_n^p} \right]^{\frac{1}{p}} \qquad (5.3)$$

For the mixed queries with both *AND* and *OR* operators, the formulas (5.2) and (5.3) can be appropriately extended. For example, for the query $Q = \{(t_1, q_1)\ OR\ [[(t_2, q_2)\ AND\ (t_3, q_3)], x\}$ and let $Q' = \{(t_2, q_2)\ AND\ (t_3, q_3)\}$, we have

$$sim(\vec{d}_\alpha, Q) = \left[\frac{a_{\alpha 1}^p w_1^p + x^p\ [sim(\vec{d}_\alpha, Q')]^p}{w_1^p + x^p} \right]^{\frac{1}{p}} ,$$

where

$$sim(\vec{d}_\alpha, Q') = 1 - \left[\frac{(1 - a_{\alpha 2})^p w_2^p + (1 - a_{\alpha 3})^p w_3^p}{w_2^p + w_3^p} \right]^{\frac{1}{p}}$$

In the p-norm model, the interpretation of the query can be altered by using different p-values to compute the query-document similarity. When p = 1, the similarity measures for the *OR*-query and *AND*-query are the same. In other words, the distinction between the *AND* and *OR* connectives in a query disappears and the conventional vector processing model, where the similarity is based on the simple matching function, is obtained.

In contrast, the case that the query terms are all equally weighted, the expressions (5.2) and (5.3) can be written as

$$sim(\vec{d}_\alpha, Q_{OR^\infty}) = \lim_{p \to \infty} \left[\frac{a_{\alpha 1}^p w_1^p + a_{\alpha 2}^p w_2^p + \ldots + a_{\alpha n}^p w_n^p}{w_1^p + w_2^p + \ldots + w_n^p} \right]^{\frac{1}{p}}$$

$$= max\ [\ a_{\alpha 1} + a_{\alpha 2} + \ldots + a_{\alpha n}] \qquad (5.4)$$

and

$$sim(\vec{d}_\alpha, Q_{AND^\infty})$$

$$= \lim_{p \to \infty} 1 - \left[\frac{(1-a_{\alpha 1})^p w_1^p + (1-a_{\alpha 2})^p w_2^p + \ldots + (1-a_{\alpha n})^p w_n^p}{w_1^p + w_2^p + \ldots + w_n^p} \right]^{\frac{1}{p}}$$

$$= min[a_{\alpha 1} + a_{\alpha 2} + \ldots + w_{\alpha n})] . \qquad (5.5)$$

Equations (5.4) and (5.5) indicate that when p = ∞ and the query is unweighted, the value of the query-document matching function is given by the highest weight among document terms for an *OR*-query and lowest weight among document terms for an *AND*-query. This is precisely the way in which weights are handled in the fuzzy set model. It is easy to see that under this special case when document terms are also unweighted, the expressions reduce to the strict Boolean model.

In comparing the unified GVSM to the p-norm model, the following points are note worthy:

(i) Both models handle weighted Boolean queries. The dependencies perceived by the user can be incorporated into the query by using Boolean connectives.

(ii) Both reduce to VSM and strict Boolean retrieval models under certain conditions.

(iii) p-norm model involves the parameter p, which has to be experimentally determined. No such parameters are involved in the unified GVSM. Moreover, the range of p-values is not as broad as it appears since there is very little difference in performance between p-values of 2 or 3 versus larger values of p [S83].

(iv) The extended query language satisfies more algebraic properties under unified GVSM (e.g. associativity is violated in the p-norm model).

In the unified GVSM, there is not a smooth transition from extended Boolean at one extreme to the conventional vector processing at the other extreme. In fact, we feel that the unified GVSM is closer to the strict Boolean model than it is to VSM. In other words, the role of Boolean operators is rather strictly retained. The p-norm model, however, achieves more *softening* of the operators.

The p-norm model has the problem that it ignores term correlations. The notion of L_p-norm is applied to obtain similarity functions $sim(d_a, Q_{OR^p})$ and $sim(d_a, Q_{AND^p})$ assuming that the documents can be represented in a n-dimensional vector space spanned by a set of *orthonormal* basic term vectors. For example, when p = 2, the length of any $\vec{d}_a$ can be expressed as

$$\|\vec{d}_a\| = (\vec{d}_a \cdot \vec{d}_a)^{\frac{1}{2}}$$

$$= \left(\sum_{i=1,j=1}^{n} a_{ai}\, a_{aj}\, \vec{t}_i \cdot \vec{t}_j \right)^{\frac{1}{2}}$$

In representing the vector length using the L_2-norm, it is implicitly assumed that terms are not correlated (or, that all the terms are pairwise orthogonal). Then $\|\vec{d}_a\|$ can be written as

$$\|\vec{d}_a\| = (a_{a1}^2 + a_{a2}^2 + \ldots + a_{an}^2)^{\frac{1}{2}}$$

Since term vectors are not in general orthogonal to each other, it is questionable whether the notion of generalized scalar product, for the similarity function, can still be used. The unified GVSM, in contrast, precisely specifies what is meant by term correlation and how it is to be incorporated into the retrieval scheme.

5.2. Experimental Evaluation

5.2.1. General specification

In the experiments performed, the following collections are used: MEDLARS and CISI. Only these two collections could be used because the other collections commonly used for IR experiments do not provide Boolean queries. MEDLARS is a collection of 1033 documents in the field of biomedicine. There are 30 queries designed to be used in the Boolean processing environment. The MEDLARS collection was prepared by the National Library of Medicine. The CISI consists of 1460 documents on library science, and has 35 Boolean queries. The CISI collection was obtained, via Cornell University,

MEDLARS (1033 DOCS, 30 QUES)				
Recall	Precision			
	strict Boolean	fuzzy-set	p-norm,p=2	unified GVSM
0.1	.5528	.6339	.7532	.6907
0.2	.4313	.4609	.6875	.6028
0.3	.3065	.3398	.6267	.5274
0.4	.2370	.2525	.5486	.4972
0.5	.1630	.1866	.4493	.4638
0.6	.1532	.1779	.4323	.3828
0.7	.1065	.1259	.3655	.3362
0.8	.0769	.0926	.3059	.2893
0.9	.0381	.0525	.1610	.2301
1.0	.0321	.0448	.0866	.1518
Average precision	.2097	.2373	.4417	.4172
average % improvement over strict Boolean		+13%	+111%	+99%

Table 5.1. Performance of different Boolean systems.

CISI (1460 DOCS, 35 QUES)				
Recall	Precision			
	strict Boolean	fuzzy-set	p-norm,p=2	unified GVSM
0.1	.2259	.2916	.3559	.2950
0.2	.1951	.2397	.2718	.2324
0.3	.1405	.1592	.2112	.2052
0.4	.1202	.1312	.1869	.1668
0.5	.0982	.1048	.1671	.1503
0.6	.0800	.0808	.1427	.1305
0.7	.0544	.0547	.1229	.1054
0.8	.0482	.0478	.0975	.0862
0.9	.0405	.0399	.0782	.0625
1.0	.0361	.0354	.0542	.0408
Average precison	.1040	.1185	.1688	.1475
average % improvement over strict Boolean		+14%	+62%	+42%

Table 5.2. Performance of different Boolean systems.

from the Institute of Scientific Information. The query collections include, for evaluating purposes, information as to which documents are relevant to each query. The standard recall and precision measures are used for comparing the performance of different information retrieval models. *Recall* is defined as the proportion of relevant documents retrieved out of the total number of relevant in the whole collection and *precision* is the proportion of the relevant documents retrieved out of the total number retrieved. The overall performance of a retrieval strategy is determined by computing the average precision for recall values 0.1, 0.2, . . . , and 1. The algorithm for averaging is consistent with that implemented in the SMART system.

5.3. Experiments with extended Boolean queries

For comparison purposes, performance results for strict Boolean, fuzzy set and p-norm systems were obtained from the Cornell University [S84, S85]. Binary weights for both documents and queries are used in strict Boolean, p-norm and GVSM models, while weighted documents are used in the fuzzy-set model runs. In particular, the results referred to as fuzzy-set correspond to the use of the p-norm model with $p = \infty$. In a strict Boolean system, where the set of relevant and nonrelevant retrieved items is not ranked by the system, a ranked output can be simulated by defining an order for the retrieved items in which relevant items are randomly placed among the set of nonrelevant ones. Such a simulated ordering operation permits recall and precision computation for the base method. For the p-norm model, the results corresponding to $p = 2$ are reported because the *AND* and the *OR* operators are treated differently in this case. Tables 5.1 and 5.2 summarize the results, respectively, for MEDLARS and CISI collections. The columns labelled strict Boolean and fuzzy-set are from Salton and Voorhees [S85]. The results of p-norm (p=2, binary weights) were obtained from Cornell by private correspondence. The improvement of fuzzy set, p-norm and unified GVSM approaches over strict Boolean for the MEDLARS collection are, respectively, 13%, 111% and 99%. When eqn.(3.4) is used to transform Boolean queries, under some special conditions, a query may be converted to a null vector. This happened for Q_8 and Q_{10} in this collection. Therefore, the unified GVSM results are averaged over only 28 queries. The performance improvements for the above methods in the case of the CISI collection are 14%, 62% and 42%. From these results, it is seen that unified GVSM provides substantial improvement over fuzzy set and strict Boolean approaches. However, these improvements are not as high as that for the p-norm model.

In order to provide a proper perspective on the results reported above, we next provide a comparison of the unified GVSM relative to the performance of the p-norm model for a range of p-values. These results are summarized is Tables 5.3 and 5.4. For comparison purposes a single precision value is used, which represents the average of precision values at three typical recall levels. The recall levels considered are a low recall of 0.25, a medium recall of 0.5 and a high recall of 0.75. In cases where precision values were known for values 0.1, 0.2, . . . , 1.0, the values for 0.25 and 0.75 are computed by averaging, respectively, precision at (0.2, 0.3) and precision at (0.7, 0.8).

In Section 5.1 it was mentioned that the operators are applied rather strictly under the unified GVSM. The results in Tables 5.3 and 5.4 are consistent with that observation. That is, the unified GVSM provides precision values that seem comparable to the situation where a p-value of around 9 is used, for both collections. Thus, the trade off one has to consider is between maintaining desirable algebraic properties (and, hence alleviate confusion during query formulation) versus permitting considerable softening of the operators. It is not difficult to convince oneself that greater softening of operators would lead to more algebraic properties being distroyed. The use of parameter p, while providing certain flexibility, brings with it the problem of having to determine the best value of p for each situation.

MEDLARS (1033 DOCS, 30 QUES)				
Type of Run	p-values	Average precision at three recall points	% difference from strict Boolean	% difference from cosine
Strict Boolean	∞	.2079*	--	-62%
p-norm Binary weights	1	.4710+	+126%	-14%
''	2	.4767+	+129%	-13%
''	5	.4720+	+127%	-14%
''	9	.4599+	+121%	-16%
Unified GVSM	--	.4473	+115%	-18%
p-norm weighted docs.	∞	.2321*	+11.2%	-58%
VSM(cosine)	--	.5473+	+163%	--

Table 5.3. Comparison of unified GVSM with p-norm model
(* -- source is [S85]; + -- source is [S83a]).

CISI (1460 DOCS, 35 QUES)				
Type of Run	p-values	Average precision at three recall points	% difference from strict Boolean	% difference from cosine
Strict Boolean	∞	.1059*	--	-32%
P-norm Binary weights	1	.1687+	+59%	+7%
''	2	.1692+	+60%	+8%
''	5	.1691+	+60%	+8%
''	9	.1586+	+50%	+1%
Unified GVSM	--	.1549	+46%	-1%
P-norm weighted docs.	∞	.1185*	+12%	-24%
VSM(cosine)	--	.1569+	+48%	--

Table 5.4. Comparison of unified GVSM with p-norm model.
(* -- source is [S95]; + -- source is [S83a]).

Although the unified GVSM does not have a parameter such as p in the p-norm model, the two extreme cases have an analogue. The case p=1 corresponds to the basic GVSM and the case $p = \infty$ corresponds to unified GVSM where documents are assumed to be orthogonal (see Section 3.1) to each other. In general, we

hypothesize that documents and queries should be represented by a vector sum of $\vec{m}$'s. The choice made for representation of documents can be varied to involve several $\vec{m}$'s (according to some criterion) or just the dominant atom as in the strict Boolean model. This perspective suggests the possibility that document representation can be varied by having other prescriptions for the choice of $\vec{m}$'s. We are currently looking at some experiments along these lines.

It is shown in Wong et al. [WO85] and Wong [W85] that the basic GVSM obtains significantly better results than VSM for almost all collections. One exception is the ADINUL collection, where there seem to be little need to be bothered with term correlations. In the light of the fact that basic GVSM obtains extremely good performance and the p-norm model does not really take term correlations into account, we believe that it is attractive to develop models that provide the benefits of both by incorporating term correlation and employing a parameter such as p to fine-tune performance according to the situation at hand.

6. CONCLUSIONS

In the context of the standard vector space model [S71, S83], there has been no formal method for determining term-term correlations or for the incorporation of such correlations into the retrieval strategy. A model termed the GVSM (Generalized Vector Space Model) was introduced to fill the gap [WO84, WO85]. Extensive experiments have been performed on the *basic* GVSM and the scheme has been found to be highly successful [W85]. The basic GVSM, however, assumes the query to involve no structure. That is, the queries are specified as a list of (term, weight) pairs. In this paper, a prescription for processing extended Boolean queries under the premises of GVSM is advanced. This extended model, termed the *unified* GVSM is compared with the p-norm model both conceptually and experimentally. The advantages and disadvantages of each are identified. The important conclusion is that it appears likely that an even more comprehensive model that combines the unified GVSM and the p-norm models can be formulated. Such a model can incorporate the positive aspects of both. Further investigations in this direction are warranted.

REFERENCES

[BU81] Buell, D., "A General Model of Query Processing in Information Retrieval Systems," Information Processing & Management, 17 (1981), pp. 249-262.

[C82] Croft, W., "Experiments with Representation in a Document Retrieval System," COINS Technical Report 82-21, University of Massachusetts (1982).

[GE82] Gersting, J. L., "Mathematical Structures for Computer Science," W. H. Freeman and Co., San Francisco (1982).

[P84] Paice, C. D., "Soft evaluation of Boolean search queries in information retrieval systems," Information Technology, Vol. 3, No. 1 (1984), pp. 33-41.

[S71] Salton, G., "The SMART Retrieval System - Experiment in Automatic Document Processing," Prentice-Hall, New Jersey (1971)

[S83] Salton, G. and McGill, M. H., "Introduction To Modern Information Retrieval," McGill-Hill, Inc., New York (1983), pp. 146-151.

[S83a] Salton, G., Fox, E. A. and Wu, H., "Extended Boolean Information Retrieval," Communications of the ACM, Vol. 26, No. 11 (1983), pp. 1022-1036.

[S83b] Salton, G., Fox, E. A. and Wu, H., "An Automatic Environment for Boolean Information Retrieval," Information Processing & Management (1983), pp. 755-762.

[S84] Salton, G. and Voorhees, E., "Automatic Assignment of Soft Boolean Operators," TR 84-608, Cornell University (May 1984).

[S85] Salton, G. and Voorhees, E., "Automatic Assignment of Soft Boolean Operators," Proc. 8th Annual ACM-SIGIR Conference on Research and Development in information Retrieval (1985), pp. 54-69.

[T76] Tahani, V., "A Fuzzy Model of Document Retrieval Systems," Information Processing & Management, Vol. 12 (1976), pp. 177-187.

[V77] Van Rijsbergen, C. J. "A Theoretical Basis for the Use of Co-occurrence Data in Information Retrieval," Journal of Documentation, Vol. 33 (1977), pp. 106-119.

[W85] Wong, P. C. N., "A generalized Vector Space Model for Information Retrieval Systems," M.Sc. thesis, Computer Science Dept., University of Regina (1985)

[WA79] Waller, W. and Kraft, D. H., "A Mathematical Model of a Weighted Boolean Retrieval System," Information Processing Mangement, Vol. 15 (1979), pp. 235-245.

[WO84] Wong, S. K. M. and Ziarko, W., "A Unified Model in Information Retrieval," Fundamenta Informaticae Vol. X (1987).

[WO85] Wong, S. K. M., Ziarko, W. and Wong, P. C. N., "Generalized Vector Space Model in Information Retrieval," Proceedings of the Seventh International Conference on Information Storage and Retrieval (1985), pp. 18-25.

An Experimental Study of Factors Important in Document Ranking

Donna Harman

Lister Hill National Center for Biomedical Communications
National Library of Medicine
Bethesda, Maryland, 20209

Abstract

The ability to effectively rank retrieved documents in order of their probable relevance to a query is a critical factor in statistically-based keyword retrieval systems. This paper summarizes a set of experiments with different methods of term weighting for documents, using measures of term importance within an entire document collection, term importance within a given document, and document length. It is shown that significant improvements over no term weighting can be made using a combination of weighting measures and normalizing for document length.

1. Introduction

There is considerable interest in retrieving information from existing sources as different as semi-structured databases and unstructured document collections, without having to re-organize the data and without requiring a complicated syntax for submitting queries. A statistically-based keyword system, using automatic indexing of the database and natural language queries, is a very attractive choice, offering ease of implementation, minimum modifications to the database, and availability to users of natural language. A major element in the success of these keyword systems is their ability to rank the retrieved items in order of probable relevance to the query. Consequently, the choice of the factors and the weightings of these factors in the ranking algorithm are critical to retrieval effectiveness. The experiments described in this paper were aimed at identifying the factors important in ranking and at examining how to most effectively combine them.

These experiments were done using the IRX testbed put into operation in 1985 at the Lister Hill National Center for Biomedical Communications, the research arm of the National Library of Medicine.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

The IRX (Information Retrieval Experiment) project was started in 1984 to answer two major questions. First, given the wealth of experimentation done using statistically-based keyword retrieval, what are the upper bounds of performance, and what new techniques can be developed to extend these bounds to those projected for expert systems. Included in this question is an estimate of the cost required to attain this higher level of performance in terms of software development, maintenance, and user effort. Second, what are the basic parameters in a traditional statistically-based keyword retrieval system, and what kinds of changes in these parameters are needed to match different document collections or parsing techniques. For example, is it possible to directly tune a retrieval system to perform effectively given a set of database characteristics or a particular parsing method?

Research is currently being done in two major areas. First, collaboration with the Welch Medical Library at the Johns Hopkins School of Medicine will result in both a new test collection and the opportunity to allow users to access a collection of medical text online. At a later stage of research there are plans to have users test intelligent interfaces that interact with the user to refine search requests. The second area involves the establishment of a baseline level of performance, and the investigation of the factors important in a statistically-based keyword retrieval system. In this area, ranking techniques have been selected as the first field of investigation, and the results of this research form the basis for this paper.

2. Current ranking techniques

Five different components in the ranking of documents have been reviewed by McGill: 1) the form of document representation, 2) the weighting of the document terms, 3) the form of query representation, 4) the weighting of the query terms, and 5) the similarity measure [MCGILL79]. Of these five components, only the weighting of the document terms was investigated in this experiment; the others were held constant. Automatic indexing using full words was used as the form of document (and query) representation, binary weighting of the query terms was used, and the similarity measure used was an inner product.

Examining the weighting of document terms raises two major questions: 1) what factors in a document are important in measuring document-query similarity, and 2) how should these factors be measured and combined. In reviewing past research, four major factors emerged: 1) the number of term matches between the query and a document, 2) the importance of a given term within a document collection, 3) the importance of a given term within a given document, and 4) the length of a document.

2.1 The number of term matches between a query and a document

The number of term matches between a query and a document is an elementary measure of document-query similarity, in which the rank of a document is based on the number of document and query terms that match. This measure, which can be viewed as a binary weighting of the document terms, has been used as a baseline for comparing different ranking methods [CROFT82]. The technique works better than no ranking, but fails to discriminate between terms of varying importance, or between documents that provide more matches just because they are

2.2 The importance of a term within the entire document collection

Many functions have been developed for measuring the importance of a term within an entire document collection. One of the first, the inverse document frequency [SPARCK JONES72;SALTON83], is based on the number of documents in which a given term k occurs in a document collection. Specifically;

$$inverse\ document\ frequency_k = \log_2 \frac{N}{NumDoc_k} + 1$$

where N = *number of documents in document collection*
$NumDoc_k$ = *number of documents in the collection that contain one or more instances of term k*

A second measure, noise, [DENNIS64;SALTON83] also measures term occurrence within a collection, but measures the concentration of that term rather than occurrence counts. Specifically

$$noise_k = \sum_{i=1}^{N} \frac{Freq_{ik}}{TotFreq_k} \log_2 \frac{TotFreq_k}{Freq_{ik}}$$

where N = *number of documents in document collection*
$Freq_{ik}$ = *the frequency of term k in document i*
$TotFreq_k$ = *the total frequency of term k in the document collection*

If a term appears only in one document, the noise is zero, but for terms that are fairly evenly distributed within a document collection, the noise is much higher.

Both inverse document frequency weight and noise were investigated in the present set of experiments. Other measures of term importance within a collection were not investigated: 1) the term discrimination measure [SALTON73], 2) the term precision value [YU82], and 3) the 2-Poisson model [HARTER75;RAGHAVAN83]. These measures involved a level of computational complexity that was beyond the scope of the current investigation.

2.3 The importance of a term within a given document

The importance of a term within a given document is usually expressed as a function of its frequency within that document. Two functions were investigated in this study: 1) the raw frequency, and 2) the $\log_2$ of frequency.

2.4 Normalization for document length

The factor of document length is not generally directly used, but becomes involved in the way the factors are combined (such as the use of a cosine similarity measure). This factor was used directly as a final normalizing function in the latter set of experiments.

2.5 Combining the measures

The various ranking factors can be combined in many ways to create a total term weight for a given document. Often, the two types of term importance factors--importance within a collection and importance within a document--are multiplied to create a single term weight [SALTON83]. This weight is used in place of a binary weight (present or not present) in one of many similarity measures [MCGILL79]. Alternatively, the various factors can be combined additively, with constants used to weight the importance of each factor [CROFT79]. Both ways of combining factors were investigated in these experiments.

3. Methodology

3.1 Experimental Methods

The test collection used was the Cranfield collection with 225 queries and 1400 documents. The documents and queries were indexed using full words and a standard common word list. The terms in the query were treated as having binary weights, and the weighting factors applied only to the document terms. The inner product of query terms and document terms was used as the similarity measure.

Using batch mode, each query was parsed into noncommon terms, and a list of all documents containing one or more of these terms was input to the ranking routine. Note that since the indexing method is constant, the same list of documents was retrieved for all ranking methods, and therefore the experiments compared only the ranking of a given set of retrieved documents.

The experiments were run in the following order:

a) investigation of the use of single measures alone in the weighting of document terms: 1) the number of term matches, 2) the within-document frequency, 3) the $\log_2$ of the within-document frequency, 4) the inverted document frequency weight measure, and 5) the noise measure;

b) investigation of additively combining the number of matches with one of the other four measures to form a term weight;

c) investigation of additively combining one of the measures of the importance of a term within a given document (within-document frequency and $\log_2$ within-document frequency) with one of the measures of the importance of a term within a document collection (inverted document frequency weight and noise) to form a term weight;

d) investigation of multiplicatively instead of additively combining the measures described in c) to form a term weight;

e) investigation of normalizing the resulting similarity value for a given document by the length of the document.

3.2 Evaluation

Three methods of evaluation are presented in the tables. First, the standard recall and precision measures are given, with the averaging based on that done by the SMART system. The percentage of improvement in average precision at three given recall levels is used to compare methods. Additionally the average rank recalls are given and the E measures are calculated for 10 and 30 document cutoffs. Further description of these evaluation methods can be found in [SALTON83]. All relevant documents not retrieved for a given query are assigned ranks of 1400, 1399, etc. The Cranfield documents that are considered "source" documents for a given query are not counted as retrieved (either relevant retrieved or nonrelevant retrieved) for that query.

4. Results

4.1 Single document-query similarity measures (Table 1)

Five single measures were run: 1) the number of matches, 2) the raw frequency of a term within a document, 3) $\log_2$ of the frequency of a term within a document, 4) the inverted document frequency weight measure, and 5) the noise measure.

The first, the number of matches (Run 1, Table 1), was used as the baseline for comparison of the other methods. Examination of the ranking of the documents illustrates the problems of this method of ranking. For example, query 12, with five relevant documents, ranks these at ranks 7, 72, 91, 480, and 483. One of the relevant documents has 4 terms that matched the query, two others had 3 matching terms, one had 2 matching terms, and the last relevant had only 1 matching term. The terms involved have a range of number of postings (the number of documents within the collection containing these terms) from 703 for the term "flow" to 15 for the term "machines", yet all terms are treated equally--documents containing only "flow" are randomly ordered within all documents containing only "machines". Moreover, documents containing multiple occurrences of a term are ranked no higher than ones containing only a single instance of the term.

The second and third measures, within-document frequency and $\log_2$ of within-document frequency, are based on the frequency of the matching terms within a document. The weighting for a document term is the frequency (or $\log_2$ of the frequency) of the term within the document being ranked. Again, with respect to query 12, the five relevant documents rank at 40, 103, 223, 351, and 965 using the raw within-document frequency of a matching term as its weight. Using the $\log_2$ of that frequency, the ranks are 28, 71, 76, 229, and 900. Comparing this performance for query 12 to that for using only the number of matches shows a decline in performance using the raw frequency, and a slight improvement for $\log_2$ of the frequency. This type of performance is reflected in the averages over 225 queries (Runs 2 and 3, Table 1). Using the raw frequency alone, the performance is 22% worse than the baseline simple matching. When the $\log_2$ of the frequency is used instead of the simple raw frequency, the performance improves slightly, showing an average improvement in precision of 6% over simple matching. The other evaluation measures verify these findings. The explanation for the performance difference lies in that fact that using the raw frequency measure alone allows terms of high frequency in a document (3 or more occurrences) to dominate the similarity measure. A term appearing 3 or 4 times should usually not have the same importance as 3 or 4 matching terms appearing once. By using the $\log_2$ of the frequency, this effect is lessened, and the within-document frequency becomes a more reasonable measure.

The fourth measure, the inverted document frequency weight, (measured by the inverse of the number of postings, see equation 1), takes the importance of a term within a document collection into account. The weighting for a document term is simply the inverted document frequency weight of the term. This weighting method improves the ranks of the relevant documents for query 12 to 4, 5, 30, 125, and 515. For example, relevant document 649 moves from rank 91 to rank 5, because two of the three matching terms have high inverted document frequency weights, based on their relatively low number of postings (31 for "ground" and 5 for "machines"). This type of improvement, although not uniformly better for all queries or documents, averages to a 20% improvement in average precision over the method using only the number of matches (see Run 4, Table 1). All other evaluation measures also show significant improvement.

The fifth measure, noise, also measures the importance of a term within a document collection, but is based on the distribution of the term throughout the collection and within each document (see equation 2). The weighting of a document term is a normalization of the noise of that term. The noise measure needs to be normalized since the importance of a term is in inverse relation to its noise. Normalization was done by subtracting the actual noise of a term from the maximum possible noise in the collection (9.43 for the full-word-indexed Cranfield 1400 collection), so the normalized noise ranges from 0.00 (very noisy term) to 9.43 (very low noise). Examining the ranking of the relevant documents in query 12, there is still more improvement, up to ranks 3, 4, 28, 73, and 383. When the words in the retrieved documents are examined, it appears

that the noise measure provides a fine tuning on the inverted document frequency weight measure by taking into account the distribution within documents, not just within the collection. For example, relevant document 194 moves from rank 515 to 363, based on the single matching word "calculated". This term has a medium number of postings (153), leading to a medium value of inverted document frequency weight. Since, however, the additional occurrences of the word tend to be concentrated in few documents, the normalized noise is relatively higher than the inverted document frequency weight, and this gives the improvement in rank. A second example in the same query is the word "flow", which has a high number of postings (703), but an even higher frequency (1854) with a very even distribution, leading to an extremely low normalized noise. This causes many non-relevant documents that had previously been ranked higher, based on the inverted document frequency weight measure, to be lowered in probable importance using the normalized noise measure. It should be noted that because of individual word idiosyncrasies, the normalized noise measure does not always improve performance on a query-by-query basis, but offers a slight improvement (4%) in average precision over the inverted document frequency measure, and a very significant improvement (24%) over the baseline method (see Run 5, Table 1). This is verified by the other evaluation measures, with the improvement being greatest at low levels of recall.

4.2 Combination I: Combining the number of matches with other single measures (Table 2)

Combining the number of matches with each of the other single measures produced interesting results. The weighting of a document term is a constant C_1 plus a second constant C_2 times the term weight generated by the measure being examined. Combining the number of matches with the two measures of frequency-within-document gives significant improvement (Runs 2 and 3, Table 2). When the number of matches was combined with the two measures of term importance within a collection (inverted document frequency weight and normalized noise), there is no significant change in performance (Runs 4 and 5, Table 2). The combining of the number of matches and the inverted document frequency weight was used by Croft [CROFT79] in working with a manually-indexed version of the Cranfield 1400, with slightly better results. As expected, if the constants C_1 or C_2 are increased from 1, performance approaches that of the measure being more heavily weighted (results not shown). These findings are verified by the other evaluation measures. The performance difference is due to the nature of the two types of measures--the two within-document frequency measures vary from 1 to the maximum (or $\log_2$ of the maximum) frequency for each term, applying additional weight for higher frequency terms. Adding the number of matches to this type of term-importance measure gives more weight to the number of matches, not just the frequency of the terms. Adding the number of matches to either the normalized noise or the inverted document frequency weight just increases each term weight by the same amount, producing little relative difference in ranking. These results suggest the reasonable combination of a frequency-within-document measure and a measure of term importance within an entire collection.

4.3 Combination II: Additively combining the single measures for term importance within a collection and term importance within a given document (Table 3)

The single measures were additively combined in a way similar to combination I. The weighting of the document terms is a constant C_1 times the first measure plus a second constant C_2 times the second measure. Only one measure for term importance within a given document was used ($\log_2$ of the within-document frequency), since using the raw frequency was shown to be a significantly worse measure. Both measures for term importance within a document collection were tried. Several types of results can be seen in Table 3. First, the effects of additively combining the two types of term importance measures is roughly additive (6.0 + 19.9 for inverted document frequency weight and 6.0 + 23.9 for normalized noise), with the normalized noise measure showing the greatest gain (Runs 5 and 6, Table 3). This is an indication of the complementary nature of the two types of measures--the effects do not mask each other. The second finding apparent from Table 3 is the continued slight superiority of the normalized noise measure--adding the within-document frequency has widened the performance difference. Third, the weighting of the two types of measures seems to be best at equal weighting, but other weightings make only slight decrements in performance (Runs 7 and 8, Table 3). These results also hold true for the inverted document frequency weight measure although the data is not included in Table 3. Adding the number of matches causes a slight, but insignificant decline in performance (data not included in Table 3). These results are verified by the other evaluation measures.

4.4 Combination III: Multiplicatively combining the single measures for term importance within a collection and term importance within a given document (Table 3)

The single measures were multiplicatively combined in a manner similar to combination II. The weighting of the document terms is the product of the first measure and the second measure. Table 3, Runs 9 and 10, shows this method of combining single measures, leading to a 29% and 35% improvement over simple matching, to be better than additive combinations. The results are consistent with those for additive combinations--adding the number of matches makes no significant improvement (data not included in the table), and the normalized noise measure still performs more effectively than the inverted document frequency measure. All these results are verified by the other evaluation measures.

4.5 Modification for length (Table 4)

Documents that are significantly longer than the average document length in the collection can have a higher rank simply because they are longer. This affects the rank both by increasing the total number of matches, and by causing higher frequencies of the matching terms. The effects of document length are usually handled indirectly by the use of a cosine similarity measure. As the

necessary document statistics were not readily available in the current IRX system, this was not possible. Additionally, directly using the factor of document length allowed a clearer insight into the effects of document length. Two measures of document length were used, analogous to the two measures of within-document frequency--the actual length of a document (as measured by the total number of characters needed to store the document) and the $\log_2$ of the actual length. The length modification used was a division of the term weight of a document by its length (or the $\log_2$ of its length). Table 4 shows the effects of using document length to modify the document weights for the additive and multiplicitive combinations of measures. As can be seen, modification by the raw document length does not improve performance over the combinations (Run 3, Table 4--other combinations not shown), but modification by the $\log_2$ of the document length improves performance very significantly, leading to a maximum improvement of 44% over simple match (Runs 4, 5, 6, and 7, Table 4). Using the $\log_2$ of the document length dampens the effect of document length, preventing severe penalties on extra long documents.

5. Conclusions/Future Research

The use of term weighting in a document produces significant gains in performance, up to a 44% improvement in average precision over simple matching. Additionally the following conclusions can be drawn from the experiments.

a) The three types of measures tested: 1) the importance of a term within a document collection, 2) the importance of a term within a given document, and 3) the length of a document, are all important in term weighting of documents.

b) The two types of term-importance factors, importance within a collection, and importance within a given document, measure term usage in two complementary places--within a given document and within an entire document collection--and combining them produces a cumulative effect.

c) The normalized noise measure of term importance within a document collection is a viable alternative measure to the inverted document frequency measure.

d) Using the $\log_2$ of the frequency of a term within a document instead of its raw frequency produces a superior measure of the importance of a term within a given document.

e) Combining the two types of term importance factors, term importance within a document collection and term importance within a given document, is more effective than using single factors alone. Combining them multiplicatively produces the best results for the test collection and indexing methods used in these experiments.

f) Adding the number of matches between a document and a query to a term weight

produced by any combination involving a factor that measures term importance within a collection does not produce significant improvement in performance, at least for this test collection and for full word indexing.

g) It is important to consider the length of a document in ranking. Dividing the total term weight by the $\log_2$ of the document length produces significant performance improvement for this test collection.

Future research is needed to resolve some of the issues. The $\log_2$ of the frequency and of the document length is only one possible function of these measures, and other moderating functions should be examined. Similarly the normalization of the noise measure should be done using alternative methods.

Additionally these experiments need to be run on other test collections, not only as verification, but to begin to investigate how factor importance varies between collections. This includes the investigation of both single factors, and the combining and weighting of these factors. Various indexing methods, such as suffixing, need to be used, with experimental work done into what changes are needed in document term weighting to best match a given indexing method. A thorough understanding of the role term weighting plays in a statistically-based keyword retrieval system is important for later more advanced research into intelligent user interfaces.

Acknowledgements

Thanks to the rest of the IRX project team, Dennis Benson, Larry Fitzpatrick, Charles Goldstein, and Rand Huntzinger, without whom there would be no research system. Also thanks to the SMART project, especially Chris Buckley, for the use of test collections and system code.

References

[CROFT79] Croft W.B., Harper D.J., "Using Probabilistic Models of Document Retrieval Without Relevance Information", Journal of Documentation, Vol. 35, No. 4, December 1979, pp. 285-295.

[CROFT82] Croft W.B., "Experiments with Representation in a Document Retrieval System", COIN Technical Report 82-21, May 1982.

[DENNIS64] Dennis S.F., "The Construction of a Thesaurus Automatically from a Sample of Text",Symposium Proceedings, Statistical Association Methods for Mechanized Documentation, 1964. (National Bureau of Standards Miscellaneous Publication 269).

[HARTER75] Harter S.P., "A Probabilistic Approach to Automatic Keyword Indexing", Journal of the American Society for Information Science, Vol. 26, No. 5, October 1973, pp. 280-289.

[MCGILL79] McGill M., Koll M., Noreault T., "An Evaluation of Factors Affecting Document Ranking by Information Retrieval Systems". Report, School of Information Studies, Syracuse University, Syracuse, New York, October 1979.

[RAGHAVEN83] Raghavan V.V., Yu C.T., "Evaluation of the 2-Poisson Model as a Basis for Using Term Frequency Data in Searching", Proceedings of the Sixth Annual International ACM SIGIR Conference on Research and Development in Informtion Retrieval, Washington, D.C. 1983.

[SALTON73] Salton G., Yang C.S., "On the Specification of Term Values in Automatic Indexing", Journal of Documentation, Vol. 29, No. 4, December 1973, pp. 351-372.

[SALTON83] Salton G., McGill M., Introduction to Modern Information Retrieval, McGraw-Hill Book Company, New York, 1983.

[SPARCK72] Sparck Jones K., "A Statistical Interpretation of Term Specificity and Its Application in Retrieval", Journal of Documentation, Vol. 28, No. 1, March 1972, pp. 11-20.

[YU82] Yu C.T., Lam K., Salton G., "Term Weighting in Information Retrieval Using the Term Precision Model", Journal of the American Association for Computing Machinery, Vol. 29, No. 1, January 1982, pp. 152-170.

TABLE 1

Run 1 -- number of matches only
Run 2 -- within document frequency only
Run 3 -- log2 within document frequency only
Run 4 -- inverted document frequency weight only
Run 5 -- normalized noise only

Run	1	2	3	4	5
Recall - Precision					
0.00	0.587	0.416	0.532	0.571	0.581
0.10	0.475	0.374	0.491	0.535	0.542
0.20	0.389	0.306	0.417	0.447	0.460
0.30	0.310	0.234	0.315	0.367	0.371
0.40	0.244	0.204	0.270	0.313	0.321
0.50	0.216	0.175	0.230	0.269	0.282
0.60	0.149	0.119	0.158	0.183	0.188
0.70	0.116	0.082	0.119	0.138	0.140
0.80	0.093	0.067	0.095	0.115	0.116
0.90	0.074	0.051	0.074	0.093	0.092
1.00	0.071	0.047	0.069	0.088	0.087
Average precision for 3 intermediate points					
Precision	0.224	0.176	0.237	0.268	0.277
% Precision Change		-21.5	6.0	19.9	23.9
Norm Recall	0.844	0.847	0.853	0.855	0.856
Precis after 10 docs	0.160	0.135	0.175	0.188	0.186
Precis after 30 docs	0.094	0.081	0.096	0.103	0.106
Recall after 10 docs	0.269	0.223	0.290	0.323	0.328
Recall after 30 docs	0.439	0.379	0.446	0.487	0.501
E, 0.5, 10 docs	0.835	0.862	0.820	0.805	0.806
E, 1.0, 10 docs	0.818	0.849	0.803	0.784	0.785
E, 2.0, 10 docs	0.784	0.821	0.766	0.742	0.741
E, 0.5, 30 docs	0.890	0.906	0.889	0.881	0.877
E, 1.0, 30 docs	0.854	0.874	0.852	0.840	0.835
E, 2.0, 30 docs	0.770	0.803	0.768	0.748	0.741
Number queries	225	225	225	225	225

TABLE 2

Run 1 -- 1*number of matches
Run 2 -- 1*number of matches + 1*within document frequency
Run 3 -- 1*number of matches + 3*within document frequency
Run 4 -- 1*number of matches + 1*log2 within document frequency
Run 5 -- 1*number of matches + 3*log2 within document frequency
Run 6 -- 1*number of matches + 1*inverted document frequency weight
Run 7 -- 1*number of matches + 3*inverted document frequency weight
Run 8 -- 1*number of matches + 1*normalized noise
Run 9 -- 1*number of matches + 3*normalized noise

Run	1	2	3	4	5	6	7	8	9
Recall - Precision									
0.00	0.587	0.469	0.442	0.551	0.538	0.575	0.572	0.586	0.587
0.10	0.475	0.432	0.399	0.513	0.501	0.540	0.536	0.547	0.551
0.20	0.389	0.357	0.333	0.429	0.422	0.453	0.451	0.461	0.467
0.30	0.310	0.275	0.250	0.330	0.329	0.366	0.367	0.373	0.376
0.40	0.244	0.238	0.214	0.279	0.279	0.308	0.311	0.319	0.324
0.50	0.216	0.205	0.182	0.244	0.240	0.263	0.268	0.279	0.281
0.60	0.149	0.143	0.128	0.167	0.163	0.177	0.181	0.188	0.187
0.70	0.116	0.106	0.092	0.127	0.123	0.135	0.135	0.142	0.141
0.80	0.093	0.085	0.073	0.104	0.099	0.111	0.112	0.118	0.117
0.90	0.074	0.066	0.057	0.084	0.077	0.089	0.090	0.095	0.093
1.00	0.071	0.062	0.053	0.079	0.072	0.084	0.085	0.089	0.088
Average precision for 3 intermediate points									
Precision	0.224	0.208	0.189	0.247	0.244	0.265	0.268	0.277	0.278
% Precision Change		-7.0	-15.6	10.2	9.0	18.7	19.8	23.7	24.5
Norm Recall	0.844	0.851	0.849	0.853	0.853	0.855	0.855	0.857	0.857
Precis after 10 docs	0.160	0.153	0.147	0.180	0.180	0.187	0.188	0.191	0.189
Precis after 30 docs	0.094	0.088	0.084	0.097	0.096	0.103	0.103	0.105	0.105
Recall after 10 docs	0.269	0.252	0.239	0.301	0.301	0.322	0.325	0.327	0.333
Recall after 30 docs	0.439	0.419	0.391	0.461	0.453	0.489	0.487	0.493	0.495
E, 0.5, 10 docs	0.835	0.843	0.849	0.814	0.814	0.806	0.805	0.802	0.803
E, 1.0, 10 docs	0.818	0.828	0.836	0.795	0.795	0.786	0.785	0.782	0.782
E, 2.0, 10 docs	0.784	0.796	0.807	0.757	0.757	0.743	0.742	0.739	0.737
E, 0.5, 30 docs	0.890	0.898	0.903	0.887	0.888	0.880	0.881	0.878	0.878
E, 1.0, 30 docs	0.854	0.866	0.871	0.849	0.851	0.840	0.840	0.837	0.837
E, 2.0, 30 docs	0.770	0.785	0.798	0.762	0.766	0.747	0.748	0.744	0.744
Number queries	225	225	225	225	225	225	225	225	225

TABLE 3

Run 1 -- number of matches
Run 2 -- 1*log2 within document frequency
Run 3 -- 1*inverted document frequency weight
Run 4 -- 1*normalized noise
Run 5 -- 1*log2 within document frequency + 1*inverted document frequency weight
Run 6 -- 1*log2 within document frequency + 1*normalized noise
Run 7 -- 1*log2 within document frequency + 3*normalized noise
Run 8 -- 3*log2 within document frequency + 1*normalized noise
Run 9 -- 1*log2 within document frequency * 1*inverted document frequency weight
Run 10-- 1*log2 within document frequency * 1*normalized noise

Run	1	2	3	4	5	6	7	8	9	10
Recall - Precision										
0.00	0.507	0.532	0.571	0.581	0.579	0.594	0.592	0.582	0.594	0.607
0.10	0.475	0.491	0.535	0.542	0.546	0.555	0.555	0.544	0.558	0.575
0.20	0.389	0.417	0.447	0.460	0.468	0.484	0.483	0.464	0.476	0.494
0.30	0.310	0.315	0.367	0.371	0.376	0.388	0.389	0.369	0.387	0.397
0.40	0.244	0.270	0.313	0.321	0.322	0.337	0.335	0.316	0.341	0.354
0.50	0.216	0.230	0.269	0.282	0.275	0.290	0.289	0.269	0.291	0.305
0.60	0.149	0.158	0.183	0.188	0.195	0.205	0.201	0.189	0.201	0.215
0.70	0.116	0.119	0.138	0.140	0.147	0.155	0.153	0.144	0.148	0.156
0.80	0.093	0.095	0.115	0.116	0.122	0.129	0.125	0.119	0.121	0.128
0.90	0.074	0.074	0.093	0.092	0.097	0.103	0.100	0.094	0.093	0.097
1.00	0.071	0.069	0.088	0.087	0.092	0.097	0.094	0.088	0.086	0.091
Average precision for 3 intermediate points										
Precision	0.224	0.237	0.268	0.277	0.277	0.290	0.290	0.275	0.288	0.301
% Precision Change		6.0	19.9	23.9	23.8	29.8	29.5	23.1	28.9	34.7
Norm Recall	0.844	0.853	0.855	0.856	0.859	0.861	0.859	0.860	0.862	0.863
Precis after 10 docs	0.160	0.175	0.188	0.186	0.199	0.201	0.195	0.198	0.203	0.210
Precis after 30 docs	0.094	0.096	0.103	0.106	0.107	0.109	0.109	0.105	0.108	0.111
Recall after 10 docs	0.269	0.290	0.323	0.328	0.336	0.342	0.338	0.339	0.341	0.356
Recall after 30 docs	0.439	0.446	0.487	0.501	0.500	0.510	0.514	0.495	0.508	0.519
E, 0.5, 10 docs	0.835	0.820	0.805	0.806	0.795	0.792	0.798	0.795	0.791	0.783
E, 1.0, 10 docs	0.818	0.803	0.784	0.785	0.774	0.771	0.776	0.774	0.771	0.762
E, 2.0, 10 docs	0.784	0.766	0.742	0.741	0.731	0.727	0.732	0.731	0.728	0.717
E, 0.5, 30 docs	0.890	0.889	0.881	0.877	0.876	0.873	0.873	0.878	0.875	0.871
E, 1.0, 30 docs	0.854	0.852	0.840	0.835	0.835	0.831	0.831	0.838	0.833	0.829
E, 2.0, 30 docs	0.770	0.768	0.748	0.741	0.740	0.734	0.733	0.744	0.737	0.731
Number queries	225	225	225	225	225	225	225	225	225	225

TABLE 4

Run 1 -- number of matches only
Run 2 -- log2 within document frequency * normalized noise
Run 3 -- (log2 within document frequency * normalized noise)/document length
Run 4 -- (log2 within document frequency + inverted document frequency weight)/log2 document length
Run 5 -- (log2 within document frequency * inverted document frequency weight)/log2 document length
Run 6 -- (log2 within document frequeńcy + normalized noise)/log2 document length
Run 7 -- (log2 within document frequency * normalized noise)/log2 document length

Run	1	2	3	4	5	6	7
Recall - Precision							
0.00	0.507	0.607	0.561	0.622	0.631	0.631	0.629
0.10	0.475	0.575	0.523	0.588	0.607	0.602	0.606
0.20	0.389	0.494	0.459	0.510	0.520	0.526	0.528
0.30	0.310	0.397	0.388	0.417	0.418	0.433	0.428
0.40	0.244	0.354	0.325	0.346	0.364	0.366	0.372
0.50	0.216	0.305	0.275	0.297	0.314	0.308	0.326
0.60	0.149	0.215	0.201	0.207	0.227	0.219	0.234
0.70	0.116	0.156	0.169	0.160	0.168	0.171	0.172
0.80	0.093	0.128	0.141	0.132	0.139	0.142	0.142
0.90	0.074	0.097	0.108	0.106	0.108	0.112	0.108
1.00	0.071	0.091	0.101	0.100	0.102	0.106	0.101
Average precision for 3 intermediate points							
Precision	0.224	0.301	0.287	0.305	0.316	0.318	0.322
% Precision Change		34.7	28.3	36.2	41.0	41.9	44.0
Norm Recall	0.844	0.863	0.860	0.862	0.866	0.864	0.866
Precis after 10 docs	0.160	0.210	0.198	0.212	0.224	0.217	0.228
Precis after 30 docs	0.094	0.111	0.109	0.112	0.114	0.113	0.115
Recall after 10 docs	0.269	0.356	0.337	0.356	0.376	0.364	0.382
Recall after 30 docs	0.439	0.519	0.508	0.527	0.532	0.529	0.535
E, 0.5, 10 docs	0.835	0.783	0.795	0.781	0.770	0.776	0.765
E, 1.0, 10 docs	0.818	0.762	0.774	0.759	0.748	0.754	0.743
E, 2.0, 10 docs	0.784	0.717	0.730	0.714	0.700	0.708	0.695
E, 0.5, 30 docs	0.890	0.871	0.873	0.870	0.867	0.869	0.866
E, 1.0, 30 docs	0.854	0.829	0.831	0.827	0.823	0.825	0.822
E, 2.0, 30 docs	0.770	0.731	0.735	0.727	0.723	0.725	0.721
Number queries	225	225	225	225	225	225	225

A NEW THEORETICAL FRAMEWORK FOR INFORMATION RETRIEVAL

C J van Rijsbergen
Department of Computing Science
University of Glasgow

ABSTRACT

A new framework based on a non-classical logic is proposed for investigating IR. The paper motivates the use of a particular conditional logic as the 'right' logic for IR. A new principle, the logical uncertainty principle, is proposed, to deal with the inherent uncertainty associated with applicable inferences.

1. PREFACE

In the last few years, I have become increasingly dissatisfied with the state-of-the-art in information retrieval. I have reluctantly concluded that the fundamental basis of all the previous work is wrong. Almost all of the previous work in Information Retrieval (including my own) has been based on the assumption that a formal notion of meaning is not required to solve the information retrieval problems. Typically, researchers have assumed that one could get by, by only considering absence or presence of word tokens in text together with counting information about the distribution of words. Although such an approach has been successful up to a point, it has become clear that further advances in the effectiveness of retrieval by such techniques are not possible. My observation is that performance based on statistical techniques has reached its theoretical limit and any attempts to achieve further improvements are a waste of time. This is not to say that systems based on these techniques are not worth building; on the contrary, they are because it is the best we have to date. But to build a new generation of Information Retrieval systems, a new theory will be needed.

2. INTRODUCTION

This paper is to be seen as describing a new theoretical framework for investigating information retrieval. The details at this stage are not completely worked out; some further details are given in Van Rijsbergen (1986). For some years now, I have felt the need to describe such a framework. It is especially important if one wants to develop information retrieval beyond the mere keyword approach. In the closing pages of my earlier book on Information Retrieval (Van Rijsbergen, 1979) I said the following: "It has never been assumed that a retrieval system should attempt to 'understand' the content of a document. Most Information Retrieval systems at the moment merely aim at a bibliographic search.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.
© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

Documents are deemed to be relevant on the basis of a superficial description. I do not suggest that it is going to be a simple matter to program a computer to understand documents. What is suggested is that some attempt should be made to construct something like a naive model, using more than just keywords, of the content of each document in the system. The more sophisticated question-answering systems do something very similar. They have a model of their universe of discourse and can answer questions about it, and can incorporate new facts and rules as they become available".

When I wrote the above passage, I had no idea that progress in that direction was going to be so slow. The main obstacles appeared to be an adequate computable model of meaning, and its use in information retrieval operations. It was argued that even if we had an appropriate semantics for text, and it could be computed efficiently, we still would not know how to use it to retrieve documents in response to requests.

I would now like to counter this objection by saying that the use of semantics comes via an appropriate logic. I am not alone in thinking this; Cooper (1978) in his book on logico-linguistics would probably make the same claim. Such a logic would be based on a formal semantics for text. The semantics would provide a limited representation of the meaning of any text but it would not be the meaning. A logic would then be interpretable in that semantics. It leaves me to say how such a logic can help in the retrieval of relevant documents. To understand this, one must think of documents as sets of sentences which are interpreted in the semantics, and think of queries as sentences too, the latter usually a single sentence. The single primitive operation to aid retrieval is then one of uncertain implication. In the extreme case, it would be logical implication, which through its interpretation in the formal semantics is logical consequence. That is, a document is retrieved if it logically implies the request. However, as we all know, documents rarely imply requests; there is always a measure of uncertainty associated with such an implication. And so, a notion of probable, or approximate, implication is needed where a plausible inference instead of a strict inference is made, and the plausibility quantified through some measure. Modelling the information retrieval process in this way goes beyond the keyword approach, and specifies, once and for all, what relationship between a document and a request is to hold to compute probable relevance. The importance of this new way of looking at Information Retrieval derives from the realisation that with such a framework, Information Retrieval can advance with new developments in formal semantics for text. Starting with a keyword analysis which is a primitive semantics, we can go on to use our logic no matter how sophisticated our semantics is. At all times, we are attempting to infer requests (treated as sentences) from statements in the documents. The inference is possible because we have an interpretation of sentences in a document, we define this interpretation and can increase its complexity at will.

It is important to realise that the above approach is similar to the one adopted in database querying and question-answering. It is similar in that in all cases, the answer is obtained through a process of logical satisfaction, i.e. looking at a common interpretation for premises and consequent. It is different in that in the case of Information Retrieval, a request is typically a closed sentence (i.e. contains no variables) and the relationship computed between a document (the premises) and the request (the consequent) is paramount i.e. if the relationship is sufficiently strong, the document is retrieved. In the case of Data Base Management Systems, a request is typically an open sentence (contains variables), the semantics giving an instantiation of the request, which is an answer.

3. CLASSICAL INFORMATION RETRIEVAL

To begin with, I would like to say what Information Retrieval is. Let us assume that there is a large store of documents on a variety of topics. A user of such a store will have a need to know certain things, things that he does not know at present. He therefore expresses his information need in the form of a request for information. Information Retrieval is concerned with retrieving those documents that are likely to be relevant to his information need as expressed by his request. It is likely that such a

retrieval process will be iterated, since a request is only an imperfect expression of an information need, and the documents retrieved at one point may help in improving the request used in the next iteration. It is important to realise that certain words in the above description are used carefully to avoid misunderstanding the idea of information retrieval.

Let us spell out the way in which the description is to be interpreted. A request for information is translated into a request for documents. The documents are assumed to contain the information, therefore the information is only retrieved indirectly. A request is an imperfect expression of a user's information need; only a user will be able to tell whether a document contains the information he is seeking. If it does contain the information sought then the document is considered relevant to the user's information need. This implies that documents are not relevant to a request; that is, identical requests submitted by two different users can be satisfied in different ways, one document may be relevant to one user and not to the other. Relevance is here connected firmly to aboutness, a document is not relevant because of its colour or shape. It is relevant because it is about the information sought.

In specifying a model for information retrieval, a small number of entities and concepts need to be defined. Superficially, this would appear to be a simple matter. The entities and concepts are document, request, property of a document, and relevance. Anyone can give commonsense definitions of these; unfortunately, what is required is a formal definition so that an Information Retrieval system can be formally specified and therefore implemented on a computer.

Let us take a document as a set of sentences. Therefore, when a document is considered for retrieval, the sentences in the document are considered individually or perhaps jointly. In considering them, one is looking for a relationship between them and the request. Such a relationship needs to be computable if the Information Retrieval system is a computer-based one. If we take a request to be a sentence then the relationship to be computed is one between a set of sentences and a single sentence. This relationship must be such that it enables one to use it to determine whether a document is likely to be relevant or not. I use 'likely' because we are assuming that relevance is user dependent and a request is an imperfect expression of an information need.

From a system's point-of-view, the computation of the relationship between document and request is central. How is one to specify this relationship? There are several ways of doing this, each one has implications for how one represents a document and a query. Ideally, one would like this representation to be separated from the relationship computation; of course, this has proved to be almost impossible. In what follows, I propose that the right representation is given by a formal semantics for text (perhaps a Montague-style semantics, see Dowty, et al, 1981). The detailed specification of a semantics will be the subject of a later paper. The relationship between a document and a request will be formalised as a logical implication to which a measure of uncertainty is attached. To motivate this 'implication' I shall give two examples in which standard Information Retrieval models are re-expressed in terms of uncertain implication.

4. BOOLEAN RETRIEVAL

It is assumed that documents are represented by index terms, or keywords, and that requests are logical combinations (using AND, OR, NOT) of these terms. A document is deemed likely to be relevant, and hence retrieved, if the index terms in the document satisfy the logical expression in the request. For example:

D_1 = [A,B]

D_2 = [B,C] A,B,C : index terms

D_3 = [A,B,C]

$Q = A \wedge B \wedge C$

D_1 : retrieved because D_1 is true implies Q is true

D_2, D_3 : not retrieved.

The index terms are, in fact, the semantics, and indexing is seen as mapping a piece of text into its formal semantics. Formally, an index term is true for a document if it occurs in the set representing the document.

Notice the use of the closed world assumption here, that the absence of an index term in a document is assumed to imply that it is false for that document. The example makes clear that the relation computed between D and Q is one of logical implication. This is a simple set-up and commonly used in practice. Unfortunately, it does not model the uncertainty of relevance.

5. CO-ORDINATION LEVEL MATCHING

Just as in the example of Boolean retrieval above, documents are assumed to consist of sets of index terms, but requests are now also sets of index terms. The relationship between a document and a request is now computed in terms of the index terms they have in common. The likelihood of relevance is taken to be directly proportional to the number of index terms shared. For example, D_1, D_2, D_3 as before,

$Q = [A,B,C]$:

$n(D_1 \wedge Q) = 2$
$n(D_2 \wedge Q) = 2$ $\qquad$ $n(.)$: number in set
$n(D_3 \wedge Q) = 3$

This relationship can be described in terms of the probability of a logical implication, so that, $n(D \wedge Q)$ is proportional to the probability of $D \rightarrow Q$. What is a probability of $D \rightarrow Q$? This firstly depends on how one interprets '$\rightarrow$'. It is not to be interpreted as the material implication: $D \supset Q$, which is the usual truth-functional connective, only false when D is true and Q is false. Intuitively, whatever the precise meaning of '$\rightarrow$', it is easy to understand that $D \rightarrow Q$, or that $D \not\rightarrow Q$. The problem is that when $D \not\rightarrow Q$ we might still want to retrieve D because of its likelihood of relevance. To model this uncertainty of relevance, we use uncertainty of implication. If we assume $P(D \rightarrow Q) = P(Q \mid D)$, then with D and Q as sets we have:

$$P(D \rightarrow Q) = \frac{P(Q \wedge D)}{P(D)} = \frac{n(Q \wedge D)}{n(D)}$$

Treating $n(D)$ as constant, we get the relationship that $P(D \rightarrow Q)$ is proportional to the level of co-ordination.

6. A CONDITIONAL LOGIC FOR INFORMATION RETRIEVAL

In re-expressing the two well known retrieval models, Boolean and Co-ordination, as examples of computation of logical implication, I have made the case (in part) that the fundamental retrieval operation is one of logical implication. Probabilistic retrieval can also be modelled as uncertain logical implication. For this the reader is referred to Van Rijsbergen (1986). This logical implication is not one of material implication, the usual truth-functional connective $A \supset B$, which is true in all cases except when A is true and B is false. To illustrate the difference between our earlier implication $A \rightarrow B$ and $A \supset B$

let me give a simple example. First, let us assume that the probability of a conditional of the form 'If A is true then B ' is a conditional probability. Now consider a die and two events, A the event 'a number less than 3 will be rolled' and B the event 'an even number will be rolled'.

Then for the two 'implications' we get:

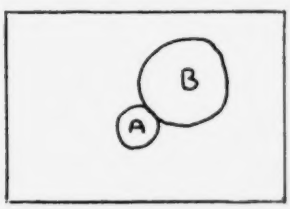
$$P(A \rightarrow B) = \frac{P(A \wedge B)}{P(A)} = \frac{1/6}{2/6} = \frac{1}{2}$$

$$P(A \supset B) = P(\bar{A} \vee B) = \frac{5}{6}$$

This shows that by interpreting the probability of a conditional as a conditional probability rather than the probability of a material implication we get widely differing results. Of course, I would maintain that the conditional probability interpretation in the context of Information Retrieval is the right one.

There is another major reason why a conditional must not be identified with the material implication in logic. When using probabilistic inference, we want to ensure that the following soundness criterion holds (Adams, 1975) : It is impossible for the premises of an inference to be probable while its conclusion is improbable. To illustrate a violation of this, we take the well known inference: Given $\sim A$ we can infer $A \supset B$. [Remember that we can logically infer a consequent from an antecedent, whenever interpretations making the antecedent true also make the consequent true.] In our example, whenever $\sim A$ is true, A will be false and hence $A \supset B$ will be true independent of B's truth value.

If we identified $A \rightarrow B$ with $A \supset B$, then such an inference could easily violate the soundness criterion. It is easy to show situations (see diagram below) where $P(\sim A)$ is large and $P(A \rightarrow B) = P(B \mid A)$ (probability of consequent) is small. In other words, although '$\sim A$ infer $A \supset B$' is valid '$\sim A$ infer $A \rightarrow B$' should not be, if we take the probabilistic soundness criterion seriously.



$P(\sim A)$ large
$P(B \mid A) \rightarrow 0$

A conditional logic will, therefore, in general, be different from a classical logic (Harper, et al, 1981). It is my contention that such a conditional logic (and there are several formulations) is the correct one for information retrieval.

7. HOW DO WE EVALUATE $P(s \rightarrow q)$?

First, let us consider the case without probabilities. To analyse this case, we will need to introduce possible-world semantics. An intuitive understanding of a possible world is that it is a complete specification of how things are, or might be, down to the finest semantically relevant details (Bradley and Schwartz, 1979). For our purposes, we will identify documents with possible worlds. This will raise problems of finiteness and structure which we will ignore for the moment.

Let s be a partial description of a document, this might be a set of sentences, or just a single index term, q will be a request. In deciding whether to retrieve a document, we would need to evaluate $s \rightarrow q$,

that is, whether $s \rightarrow q$ is true or not. If s is true in a document d then $s \rightarrow q$ is true providing q is true. If s is not true in a document then we go to the nearest document d' to d in which it is true and consider whether q is true. If q is true in d' then $s \rightarrow q$ is true in d, otherwise it is false.

To give a simple example, s might be an index term, q the same or a different index term. If $s = q$, $s \rightarrow q$ is true follows trivially for those documents in which q occurs. The more interesting case is when $s \neq q$. In this case, to establish $s \rightarrow q$ in d find the nearest document d' in which s occurs and check for the occurrence of q. It is important to realise that because of the primitive nature of the semantics, an example such as s = FORTRAN, q = PROGRAMMING LANGUAGE for which $s \rightarrow q$ is directly true in a more complex semantics, can only be handled indirectly.

The above process illustrates what is now widely known as the Ramsey test (Mellor, 1976). It might be summarised as follows:

> To evaluate a conditional, first hypothetically make the minimal revision of your stock of beliefs required to assume the antecedent. Then, evaluate the acceptability of the consequent on the basis of this revised body of beliefs.

Note the meaning of a conditional is not truth-functional under the above interpretation, i.e. its truth does not simply depend on the truth valuation of s and q in one world. It has become an intensional notion.

In document retrieval, we are often faced with the situation where $s \rightarrow q$ is assumed false because s does not logically imply q. That is, assuming the truth of the sentences (index terms) in a document we cannot arrive at q. Boolean retrieval is an excellent example: given a truth valuation for the terms describing a document, we retrieve those documents which imply q (make q true for that valuation). What is suggested here, is that a given document should be revised in a minimal way that makes s true. If, after that revision, q is true, then $s \rightarrow q$ is true and d should be retrieved. There are a number of ways of making this revision. One could restrict the revision to selecting a nearest document in which s is true, in which case, no interaction from the user would be required. Or, one could involve the user in expanding the information contained in the document under consideration. Or, finally, one could do document expansion automatically using information already stored in the system. We will return to this notion of minimal revision when we attempt to formalise it.

Turning now to the probabilistic case, to evaluate $P(s \rightarrow q)$, we revise the probability function P to P' in a minimal way, so that $P'(s) = 1$. We then have that:

$$P(s \rightarrow q) = P'(q) .$$

An example of such a revision is to make $P(s \rightarrow q) = P(q \mid s)$. In the case of Boolean semantics, where x,y are index terms and v a truth valuation:

$$v(x) = \begin{cases} 0 \\ 1 \end{cases} \qquad\qquad v(y) = \begin{cases} 0 \\ 1 \end{cases}$$

we get

$$P(x \rightarrow x) = 1$$
$$P(y \rightarrow x) = P(x \mid y)$$

In other words, a query consisting of the index term x, is related to a document containing y by $P(x \mid y)$. If we restrict our worlds to documents already present, then we can interpret this as:

$$\frac{n(x \wedge y)}{n(y)}$$

the frequency of the co-occurence of x and y divided by the frequency of y.

Of course, documents and queries are far more complex than is assumed above. It is not clear yet how one deals with arbitrary complex documents and queries. Generalising from the simple index term approach we would need to specify a formal semantics in which documents and queries would be interpreted. To evaluate $s \longrightarrow q$ would require a change in the interpretation function so that s would be true under the new interpretation, and $s \longrightarrow q$ true, if q was true as well.

8. LOGIC OF UNCERTAINTY

In evaluating the truth of $y \longrightarrow x$ or evaluating $P(y \longrightarrow x)$, we are dependent on a notion of nearness (closeness) between worlds or documents. It is interesting to examine this in a little more detail. Remember our prime concern is to establish that '$y \longrightarrow x$', or that $y \longrightarrow x$, with sufficiently large probability. If for the current document $y \not\longrightarrow x$, we look at the effect of changing/revising our current world and look at $y \longrightarrow x$ in the revised world. These changes are to be made in a minimal way.

There is another way of looking at this revision process which may be more appropriate in the Information Retrieval context. I would like to generalise the Ramsey test and state a new principle.

Logical Uncertainty Principle

> Given any two sentences x and y; a measure of the uncertainty of $y \longrightarrow x$ relative to a given data set, is determined by the minimal extent to which we have to add information to the data set, to establish the truth of $y \longrightarrow x$.

This is a slight generalisation of the foregoing. It denies that one can assess $y \longrightarrow x$ with certainty if one has to revise the data set. It says nothing about how 'uncertainty' or 'minimal' might be quantified. It specifically relativises truth to a given data set. The semantics of the data have been left unspecified too. Nearness has been replaced by a measure of information. How this measure might be evaluated is suggested in Van Rijsbergen (1986).

9. CONCLUSION

In this paper, I have given a new framework for Information Retrieval based on non-standard logic. The fundamental primitive operation relating documents and queries is taken to be logical implication. This is not a truth functional notion in the classical sense, but rather can only be evaluated by considering truth in other possible worlds. A new logical uncertainty principle is stated to characterise uncertainty associated with any logical implication; thereby quantifying the uncertainty of relevance.

ACKNOWLEDGEMENTS

I would like to thank Bruce Croft for his helpful comments in writing this paper.

REFERENCES

Adams, E.W. (1975) The logic of conditionals, Dordrecht: Reidel.
Bradley, R. and Schwartz, N. (1979) Possible Worlds, Oxford: Basil Blackwell.
Cooper, W.S. (1978) Foundations of Logico-Linguistics, Dordrecht: Reidel.
Dowty, D.R., Wall, R.E. and Peters, S. (1981) Introduction to Montague Semantics, Dordrecht: Reidel.
Harper, W.L., Stalnaker, R. and Pearce, C. (1981) (eds) Ifs, Dordrecht: Reidel.
Mellor, D.H. (1976) (ed) Foundations: Essays in Philosophy, Logic, Mathematics and Economics: F P Ramsey, London: R.K.P.
Van Rijsbergen, C.J. (1979) Information Retrieval, 2nd edition, London: Butterworths.
Van Rijsbergen, C.J. (1986) A non-classical logic for information retrieval, The Computer Journal (in press).

USER-SPECIFIED DOMAIN KNOWLEDGE FOR DOCUMENT RETRIEVAL

W.B. Croft

Department of Computer and Information Science
University of Massachusetts, Amherst, MA. 01003

Abstract

The introduction of domain knowledge into a document retrieval system has two important consequences; an increase in the effectiveness of retrieval and a decrease in the efficiency of text processing. In this paper, a method is presented of combining user-specified domain knowledge with efficient retrieval techniques based on probabilistic models. The domain knowledge is represented as a collection of frames that contain rules specifying recognition conditions for domain concepts and relationships between concepts. The inference network represented in these frames is used to infer the concepts that are related to a user's query. This approach is being implemented as part of the I^{3}R expert intermediary system.

1 Introduction

In many experimental document retrieval systems, a strong emphasis has been placed on the development of domain-independent techniques for indexing and retrieval. This emphasis arose from the desire to build systems that could, with little human intervention, store and retrieve large numbers of documents in different application areas. The systems that have been developed, such as those based on probabilistic models of relevance [RIJS79], capture "domain knowledge" purely in the statistics of occurrence of individual words (or stems) in the documents and in statistical dependencies that exist between words. We define domain knowledge to mean information about the important topics or concepts in a particular domain and how they relate to each other. The statistical approach has many advantages and can achieve a reasonable level of effectiveness with techniques that are very efficient. However, it appears that to achieve significant improvements in retrieval effectiveness compared to current techniques, systems must be designed to acquire and use explicit domain knowledge. A simple example of the use of domain knowledge is discussed in [CROF86a]. In this study, information acquired from the users about the important phrases in the queries was used to specify word dependencies for a probabilistic retrieval strategy. This extra information led to significantly better document rankings than those obtained using a strategy that treated the words as independent of each other.

Domain knowledge has of course been used for some time in the form of a thesaurus. In commercial systems, this information is used for indexing the documents and expanding the query vocabulary. Salton has conducted retrieval experiments that indicated that significant performance benefits could be obtained by using a thesaurus in automatic retrieval strategies [SALT68]. Natural language processing systems also rely heavily on domain knowledge to produce an unambiguous semantic representation of sentences. Similar techniques have been proposed for document retrieval systems [DEJO83,GUID83,BARB84]. The major drawback with these methods of using domain knowledge is that they typically assume that all knowledge is available at the time when documents and queries are processed ([BARB84] is an exception). Very little attention is paid to the acquisition of the information during the operation of the system and to the processing of documents and queries when only partial domain knowledge is available. In terms of domain independence, this represents the opposite extreme to the systems based on statistical techniques and results in systems that are limited to very specific domains or require impractical investments of time in building a domain knowledge base.

The approach to the definition and use of domain knowledge that is described in this paper addresses the problems mentioned in two ways;

1. Domain knowledge is used to enhance statistical retrieval strategies. This ensures that the system retains the benefits and baseline performance of statistical techniques.

2. Domain knowledge is acquired from the users. This knowledge is assumed to be incomplete and dynamic. This approach emphasizes the importance of individual descriptions of a domain rather than a single global description. By concentrating on domain knowledge that is relevant to particular queries, the language analysis problem is also made more tractable (similar to the approach taken in [SPAR84]).

The domain knowledge is used in the I^{3}R system [CROF86b], which is described in more detail in the next section. The system is designed to act as a search intermediary. This means that the tasks of the system include acquiring domain knowledge from the users, using this knowledge to refine query descriptions, initiating the appropriate search strategies, assisting the user in evaluating the output, and reformulating queries. It

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

is our contention that an intermediary system can provide effective retrieval with an incomplete and relatively simple domain knowledge base. The representation of domain knowledge as concept frames is discussed in the third section of the paper.

In the initial version of the system, the domain knowledge is used to infer concepts that are related to those mentioned in a particular query. The inference mechanisms that are used are based on the frame representation of concepts and the propagation of "certainty" values similar to those used in the Rubric system [TONG85]. These mechanisms are the main subject of this paper and are discussed in section 4. The concepts that are identified are used to provide information about term dependencies to the probabilistic strategies, as described in [CROF86a]. As with other systems based on probabilistic models, the representation of the contents of documents is at the level of single words. Retrieval experiments have shown that this approach will definitely improve retrieval effectiveness.

The final section of the paper discusses extensions of this work that are based on natural language processing techniques.

2 The System

The I^{3}R system (Intelligent Interface for Information Retrieval) provides assistance to users at all stages of the retrieval process. The major components of this system, which is based on the Hearsay-II speech understanding system [ERMA80], are shown in Figure 1. Each of the system experts is responsible for a specific part of the system's operation. The experts operate independently and communicate by means of a shared data structure called the blackboard. The operation of the experts is coordinated by a scheduler. The scheduler follows plans based on the actions of human intermediaries and allows the system to change its operation based on user commands or the development of a particular search session [CROF86b].

The knowledge base contains the documents and the representations of their contents. The system supports both probabilistic retrieval strategies and browsing as a means of accessing the documents. The knowledge base also contains models of individual users and session histories. The domain knowledge in the system is represented in the user models and in a "thesaurus". Domain knowledge from individual user models can be made available to all users by making it part of the overall thesaurus. The concepts described by the domain knowledge are indirectly connected to the documents through the words that form the document representatives.

The current set of system experts is as follows;

1. The **User Model Builder** collects information about the user, including the domain knowledge.
2. The **Request Model Builder** constructs a model of the user's current information need.
3. The **Indexing Expert** performs the initial processing of the document and query texts.
4. The **Thesaurus Expert** uses the domain knowledge collected from the users to identify concepts related to the query.
5. The **Search Controller** selects search strategies for a particular request.
6. The **Browsing Expert** provides an informal way of searching the documents by allowing the user to examine the contents of the knowledge base.
7. The **Explainer** explains the system's actions and knowledge to the users.

The experts that are especially concerned with the domain knowledge are the user model builder and the thesaurus expert. The browsing expert will also display this knowledge and make it available to the user for browsing. This paper concentrates on the inference mechanisms used by the Thesaurus expert.

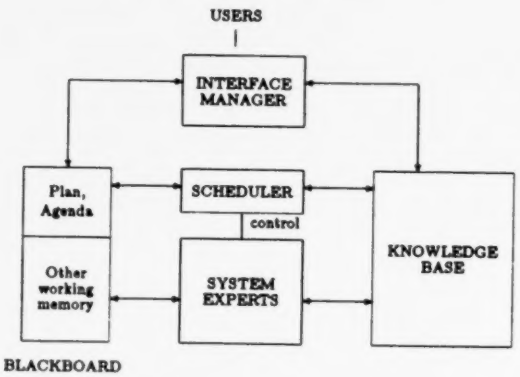


Figure 1: Major System Components.

The basic architecture of the I^{3}R system has been implemented although some of the experts are in skeleton form. A scheme for representing and manipulating plans in the scheduler is currently being tested. The user interface for specification of domain knowledge has not yet been implemented.

3 Representing Domain Knowledge

The description of a domain concept in the I^{3}R system consists of three parts;

1. The name of the concept.
2. Information about how the concept can be recognized in text.
3. The relationships between the concept and other concepts.

The three parts of a concept description are packaged into a *concept frame* and can readily be represented in any of the frame-based languages currently available [FIKE85]. This representation combines and generalizes aspects of the representations used by Shoval [SHOV81] and in the Rubric system [TONG85].

The first part of a concept description is self-explanatory. The second and third parts are more complicated and are expressed with rules that are used by the inference mechanism described in the next section. The rules in the second part of the concept frame describe the different ways in which a concept can be recognized in a text passage. The simplest form of this rule is

If "stem" *Then* Concept (C).

This rule recognizes Concept with certainty C if "stem" occurs in the text. Certainty values, which are similar to conditional probabilities, express the degree of confidence the user has in the rule and are between 0 and 1. The default certainty for recognition rules is 1. Rules of this type are used in most single-word concepts and they form the main connection between the stem-oriented document representatives generated in I^{3}R and the concepts represented in the domain knowledge. An example of this rule is

If "informat" *Then* Information (.9).

The next type of rule allows specific endings to be specified for a stem. For example,

If Ending("informat") = "ion" *Then* Information (1).

Both of these rules for the concept Information could be included in the concept frame. In general, there can be multiple rules for recognising a concept specified in the concept frame. How these rules are used depends on the inference mechanism described in the nest section.

The next type of rule refers to the presence of other concepts. This is the usual recognition rule for more complex concepts. For example,

If Information *And* Retrieval *Then* Information_Retrieval (.8).

This rule says that if the concepts Information and Retrieval can be identified in the text, the concept Information_Retrieval is recognised with certainty .8. Additional constraints can be put on the occurrence of the component concepts. For example, the rule

If adjacent(Information,Retrieval) *Then* Information_Retrieval (.9)

specifies the certainty of Information Retrieval if the concepts Information and Retrieval are adjacent in the text. Adjacency could be defined in a variety of ways.

It is interesting to note that although these rules are used for the recognition of concepts, they could also be used to generate the text strings that represent the concepts. The specification of the rules in the concept frames include pointers to any concept frames that are mentioned. Similarly, every concept frame that is mentioned in a rule contains a pointer to the frame containing that rule. These pointers are used to navigate through the frames as described in the next section.

The recognition rules mentioned so far have used very simple textual information; the presence of stems and some positional information. The same framework can also be used to specify recognition rules that use syntactic information. For example, by specifying that two concepts had to occur in the same noun phrase, this type of rule could be used to identify that Information_Management as a concept should be recognised with high certainty if the text contains "the management of information" but not if it contains "the management requires information". Processing syntax rules would of course, require a parser as well as the word level processing used in the simpler rules. Syntax rules can also be used to recognise concepts that had not previously been described. Concept frames with unspecified concept names could contain the rules that specify general syntactic criteria for concepts. For example, a rule that noun phrases represent concepts in a query could be used to identify potential concepts even if no domain knowledge had been specified. A more sophisticated approach to concept recognition in natural language queries that used some semantic information as well as syntactic information is described in [SPAR84]

The third part of the concept frame describes the relationships of a concept to other concepts. The types of relationships that are used are;

1. *Synonym.* This relates concepts that describe essentially the same thing.
2. *Generalization.* This represents the broader/narrower term hierarchy found in a thesaurus. The generalisation relationship is not the same as the "*Is-A*" link that is found in all frame-based systems because no inheritance of properties is specified.
3. *Instantiation.* The concept is an instance of another concept. For example, VAX is an instance of Computer.
4. *Part-of.* The concept is a component of another concept. This is not the same as the recognition rules. An example is that a Processor is part of a Computer.
5. *Cross-Reference.* This is a catch-all relationship to indicate that a concept has some similarity to another concept.

These relationships are represented as simple rules that identify the type of relationship, point to the related concept, and attach a certainty value to the relationship. Each of the relationships is bi-directional in that a related concept will contain a rule specifying the reverse relationship. Figure 2 summarises this section by presenting some domain knowledge represented as concept frames. In this figure, only some frames are shown in detail. Recognition rules show only the recognition condition and the certainty value because the concept name is given by the frame that contains the rule.

It is important to stress again that the domain knowledge is largely derived from system interaction with the users. This means that when a user specifies a query, the system enters a dialogue with the user with the aim of identifying concepts. The system can provide certain kinds of default information to help build the domain knowledge (for example, certainty values), but the user must contribute to this process. The I^{3}R system mimics the situation where a human intermediary interviews a user to clarify a query and build up background knowledge. If a user is not prepared to provide any domain knowledge, the system can still retrieve documents using statistical strategies, but retrieval effectiveness will suffer. Research concerned with automatically identifying domain concepts is described in the last section.

4 Using Domain Knowledge

Given a particular query and assuming that some domain knowledge is available, the I^{3}R Thesaurus Expert will use the domain knowledge to infer concepts that are related to the query. The recognition rules and relationships in the concept frames define an AND/OR tree (or goal tree) of concepts [CHAR85]. The AND nodes in the tree represent the recognition rules in that a concept represented by an AND node is recognised (or inferred) if each of the concepts represented by the children is recognised. More complex recognition rules put additional constraints on the recognition but every child node must be recognised. The OR nodes, which represent the relationship rules, can be inferred

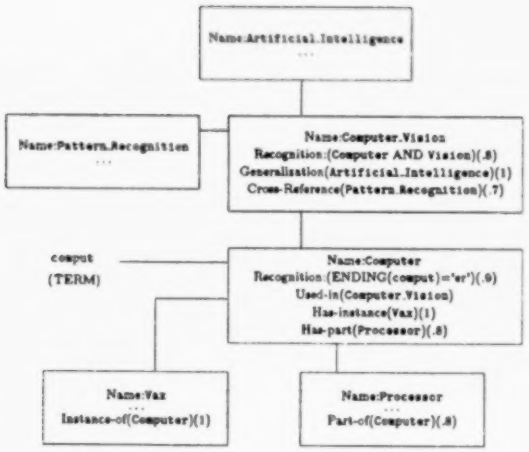


Figure 2: Domain knowledge as concept frames

whenever any child concept is inferred. Figure 3 gives an example of the AND/OR tree for the concept frames of Figure 2. The inferences represented by the relationship rules are bi-directional and, therefore, the inferences only form a tree from the point of view of a particular concept. For example, if Computer_Vision was inferred from the concepts Computer and Vision, the concept Pattern_Recognition could be the next inference.

The certainty values associated with rules are used to derive certainty values for inferred concepts. These values are provided by users or by defaults that are related to the type of rule. For example, the default certainty for a synonym relationship is .9, for a cross-reference it is .6. Certainty values can be combined

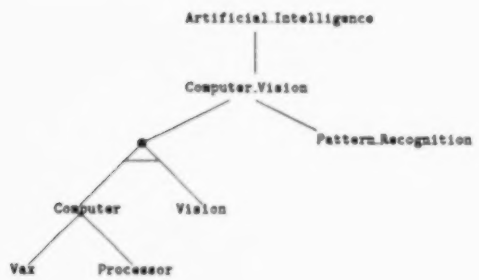


Figure 3: An AND/OR tree of concepts

in a variety of ways [TONG85]. There are two cases; inferences involving AND nodes and OR nodes. We shall illustrate the combination of certainty values using simple examples of these inferences. A simple rule that is associated with an AND node is

If A *And* B *Then* C (X).

In this rule, A,B, and C are concepts and X is the certainty associated with the rule. A and B can have their own certainty values associated with them, represented as c(A) and c(B). The overall certainty associated with the inferred concept C is defined as

$$c(C) = min[c(A), c(B)] \cdot X$$

For the corresponding OR rule, the certainty is

$$c(C) = max[c(A), c(B)] \cdot X$$

The combination of certainty values when any number of concepts are involved in the inference takes the same form.

Using the AND/OR tree as a basis, we can describe the inference process used in I^3R. The process uses two lists; a list of the concepts inferred so far and an agenda of inferences that remain to be done. Each entry on the concept-list contains the name of the concept that has been inferred, the certainty value and the chain of inferences that was followed to that concept. The latter information is used to explain the inference of a concept to the user.

The first step in the inference process is to use the stems from the query text to locate the concept frames that can be directly recognized from them. This set of concepts is placed on the concept-list. The Thesaurus Expert then tries to infer all concepts that are directly related to this initial set. In terms of the AND/OR tree, this means that the system attempts to move one level up the tree from the initial nodes (which are usually terminal nodes).

The first inferences that are done are those involving the OR nodes (concept relationships). These inferences can be made directly and are put on the concept list. The only relationship types that are used at this stage are synonym, cross-reference and instantiation in the direction of the general concept. In Cohen's terms [COHE85], the Thesaurus Expert uses only the credible inferences at this stage. As each concept frame on the initial concept list is examined, the inferences that involve AND nodes are put on the agenda. From the example in Figure 3, if Computer was a concept established from the query, the inferences involving VAX and Processor are not done and the inference involving Computer_Vision is put on the agenda.

After examining each of the initial set of concepts, the Thesaurus Expert attempts to infer the concepts on the agenda. This involves looking at the child concepts of the AND node and attempting to infer them. This process of working backwards from a goal concept is called backward chaining. By combining forward and backward chaining, the system directs the inference process towards establishing the concepts that are closely related to the query. In our example, the Thesaurus Expert would attempt to infer Computer Vision by inferring Computer and Vision. Computer is present on the concept-list and has been inferred. The Vision concept frame is then examined to find what concepts can be used to infer Vision.

In some cases, concepts that are required for recognition of an AND node concept cannot be inferred from the query. This also means that the AND node cannot be inferred. However, to allow more flexibility, a small non-zero certainty can be attached to the missing concept that will allow the AND node concept to be

inferred with low certainty. It is possible that the certainty value attached to the missing concept could be changed according to the collection frequency of the words involved.

After attempting to infer the concepts on the agenda, the system presents the concept-list to the user in order of decreasing certainty. A threshold value can be used to prevent showing concepts that are not strongly related to the query. The concepts identified as being of interest are used to refine the model of the information need (the request model). If the user requires more suggestions or if the system considers that not enough concepts have been identified, the inference process is carried further up the AND/OR tree. In this phase, the different types of rules can also be used in more specific ways. For example, if a concept is identified as very general by having a high frequency of occurrence in the document collection, the system will attempt to find more specific concepts using generalization, instantiation and part-of relationships.

5 Probabilistic Retrieval Using Concepts

The concepts derived from the query test by the inference mechanism described in the last section specify important word dependencies. These dependencies are used in a retrieval strategy based on the probabilistic model described in [CROF86a]. This model shows that documents should be ranked according to the score

$$\sum W(x_i) \cdot x_i + A \qquad (1)$$

where x_i is the ith term in the document representative, $W(x_i)$ is a weight related to the collection frequency of term i, and A is a correction factor that depends on the presence of dependent groups of words in a particular document. The summation is done over all the terms in the query. The retrieval experiments carried out with this strategy produced significant improvements when the dependencies identified were based on the user's domain knowledge.

Given that the domain knowledge can be used to find concepts related to the query text, it seems reasonable to use the same process with document texts. In this case, the inference mechanism can be used to recognise concepts that have been established during the query formulation process as part of the request model. For example, if the concept `Information_Retrieval` has been established as part of the request model, the inference mechanism can be used to infer a certainty for the concept given a particular document. This process allows query-document comparisons to be done at a higher level than the single word stems used in these systems can still be used with concepts simply providing additional "features" in the query and document representations. The certainty values derived for concepts in a particular document can be incorporated directly into the probabilistic model as a measure of the "term significance" for that document [CROF81]. More precisely, this form of the probabilistic model specifies that documents should be ranked according to the score

$$\sum T(x_i) \cdot W(x_i) \cdot x_i \qquad (2)$$

where $T(x_i)$ is the term significance weight of term i. $T(x_i)$ depends on the importance of term i in a particular document. When domain knowledge is used to identify concepts, some of the terms in expression (2) will represent concepts. For these concepts, the weight W is generally not available and must be estimated from the occurrence characteristics of the component words. The weight T corresponds to the certainty derived for the concept in a particular document. For practical reasons, it may be necessary to use the part of score (2) that comes from the concepts as a correction factor for those documents that are already highly ranked using single word stems. Since the correction factor will depend on the presence of high-level concepts in individual documents, it can be seen that this is similar to the approach used in score (1). However, it has the advantage of allowing concepts to be inferred in more sophisticated ways than word occurrence. The retrieval effectiveness of this approach has not been tested, but it would appear that, at worst, it will work as well as that based on score (1).

6 Future Work

When a user specifies a query in natural language, more information is supplied than the concepts in a particular domain. In particular, the concepts mentioned in the query sentences are related in specific ways to each other by the structure of the sentence. for example, the queries "how is vision implemented in computer systems" and "I am interested in visions of computer systems" represent very different information needs. To extend the approach described in this paper, a definition of concepts and the relationships between them needs to be made in linguistic terms. For example, Sparck Jones and Tait [SPAR84] define concepts in terms of components of a parse tree that represents the deep structure of a query. Based on such a definition, natural language processing techniques can be developed to identify the concepts and relationships in query texts. The NLP techniques would provide a robust method of identifying concepts for further inference involving domain knowledge. They would also be used to refine the matching process with document texts and to identify potential concepts in document texts. This can be seen as an extension of the process of identifying concepts in queries using syntax rules. Similar research in natural language processing has been described in [KOLO83,LEBO83].

It is hoped that by exploiting the user as the ultimate source of domain knowledge, techniques can be developed that will be practical for large collections of documents. Many other questions remain to be answered about this approach to document retrieval. These include

1. How much language processing is required to produce the query representation?
2. Where do the lexicons required for NLP come from?
3. Does the query structure give useful information about the contents of documents?
4. Is the extra effort of processing required for this approach justified in terms of performance benefits?

Although answers to some of these questions exist (for example, [ALSH85] discusses the lexicon issue), much more research needs to be done.

Acknowledgments

The author wishes to acknowledge J. Ganapathi for his work on an earlier version of the domain knowledge representation. This research was supported by NSF Grant IST-8414486. Additional facilities were provided by the Computer Science Department, University College, Dublin.

References

[ALSH85] Alshawi, H.; Bogurаev, B.; Briscoe, T. "Towards a Dictionary Support Environment for Real Time Parsing". Technical Report, Computer Laboratory, University of Cambridge, 1985.

[BARB84] Barbi, E. et al. "A Conceptual Approach to Document Retrieval". Proceedings of the ACM SIGOA Conference on Office Information Systems, 219-226; 1984.

[CHAR85] Charniak, E.; McDermott, D. *Introduction to Artificial Intelligence*. Addison-Wesley, Reading, MA; 1985.

[COHE85] Cohen, P. et al. "Representativeness and Uncertainty in Classification Systems". *AI Magazine*, 6: 136-149; 1985.

[CROF81] Croft, W. B. "Document Representation in Probabilistic Models of Information Retrieval". *Journal of the American Society of Information Science*, 32: 451-457; 1981.

[CROF86a] Croft, W. B. "Boolean Queries and Term Dependencies in Probabilistic Retrieval Models". *Journal of the American society for Information Science*, 37: 71-77; 1986.

[CROF86b] Croft, W. B.; Thompson, R. "I^3R: A New Approach to the Design of Document Retrieval Systems". Technical Report, Computer and Information Science Department, University of Massachusetts, Amherst; 1986.

[DEJO83] DeJong, G. "Artificial Intelligence Implications for Information Retrieval". Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval, *SIGIR FORUM* 17: 10-17; 1983.

[ERMA80] Erman, L. et al. "The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty". *ACM Computing Surveys*, 12: 213-253; 1980.

[FIKE85] Fikes, R.; Kehler, T. "The Role of Frame-based Representation in Reasoning". *Communications of the ACM*, 28: 904-920; 1985.

[GUID83] Guida, G.; Tasso, C. "IR-NLI: An Expert Natural Language Interface System". Proceedings of the Conference on Applied Natural language processing, 31-38; 1983.

[KOLO83] Kolodner, J. "Indexing and Retrieval Strategies for Natural Language Fact Retrieval". *ACM Transactions on Database Systems*, 8: 434-464; 1983.

[LEBO83] Lebowitz, M. "Intelligent Information Systems". Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval, *SIGIR Forum*, 17: 25-30; 1983.

[RIJS79] Van Rijsbergen, C. J. *Information Retrieval*. Second Edition. Butterworths, London; 1979.

[SALT68] Salton, G. *Automatic Information Organization and Retrieval*. McGraw-Hill, New York; 1968.

[SHOV81] Shoval, P. "Expert/Consultation System for a Retrieval Database With Semantic Network of Concepts". Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval, *SIGIR Forum*, 16: 145-149; 1981.

[SPAR84] Sparck Jones, K.; Tait, J. I. "Automatic Search Term Variant Generation". *Journal of Documentation*, 40: 50-66; 1984.

[TONG85] Tong, R. M. et al. "RUBRIC - An Environment for Full Text Information Retrieval". Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval, 243-251; 1985.

IOTA : A FULL TEXT INFORMATION RETRIEVAL SYSTEM

Y.CHIARAMELLA, B.DEFUDE, M.F BRUANDET, D.KERKOUBA

Group : Systèmes intelligents de Recherche d'Informations
Laboratoire Génie Informatique de Grenoble (IMAG)
B.P 68 38402 SAINT MARTIN D'HERES CEDEX
FRANCE

ABSTRACT

IOTA is a prototype of an Information Retrieval System which can manage a corpus made of highly structured, full text documents. The first version presented here has intelligent capabilities related to heuristic pattern matching procedures for processing natural language queries, which involve an automatically built thesaurus. The paper emphasizes the overall principles of query processing and gives hints about the underlying techniques used while constructing the thesaurus and automatically indexing highly structured documents.

1. INTRODUCTION :

IOTA is the first prototype version of an Information Retrieval System which integrates most of the various tools we designed and tested during the last few years. Although incomplete according to our objectives (hence its name!) it was of prime importance for us to check what could happen while putting together the most important pieces we developed separately..

It should be clear that IOTA is a query system; the other main functions such as automatic indexing, elaboration of the thesaurus are still separate.

This paper begins with a short presentation of underlying methods in automatic indexing and thesaurus construction, which are of prime importance to understand the query processing. For more details about chapter 2 and 3, one could refer to [KER 84] [KER 85] for automatic indexing, and to [BRU 85] for thesaurus construction. Then the chapter 4 gives detailed informations about the main features of the query system which are the following :

- access to structured texts, which allows high precision answers,
- natural language interface, though incomplete at the moment, it has proved good efficiency. This important feature is based on a syntactic pattern matching between user's concepts and system known concepts.
- relevance evaluation, which allows ordering of answers.
- utilities and user help.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

Chapter 5 concludes on the forecoming developments of this first prototype, mainly toward the improvement of intelligent capabilities via the expert system we are developing [DEF 85].

2. KNOWLEDGE REPRESENTATION - INDEXING TERMS :

2.1. Thesaurus :

At the moment we have a somewhat restricted view about the thesaurus content : central items are restricted to those effectively found in the documents and semantic relations are restricted to contextual dependencies.
For practical efficiency, we consider only the knowledge contained in all the texts of the corpus. This knowledge is represented and stored as a network of associations between words which stand for **Concepts**. So we will use the term of **partial knowledge model** rather than **thesaurus** because of its rough elaboration and its lack of complete information.
The technique used to extract concepts from corpus is based on the search and weighting of associations between words. The association strength between every pair of words is evaluated from :

- their contextual proximity (i.e lexicographical separation) in the same sentence,
- their grammatical category,
- their co-occurrence frequency.

The assumption behind this strategy is that two words close in the textual expression and frequently co-occurring are semantically bound.

The implementation of this technique is still linguistically unsophisticated (the morphological analysis is semi-automatic). The lemmatized forms of terms are generated by a program (several grammatical variants are reduced to a canonical form, for example, the infinitive for verbs, singular masculine for the adjectives and nouns). A pseudo-syntactical criteria is taken by defining a threshold for the distance between two lexical classes : for example, associations between two prepositions are not considered, and an adjective and a substantive are considered to be syntactically related only if there is a distance (i.e number of words) between them less than 9. The details of the measure used for associations and some results obtained from a technical document of 150 pages are described in another paper [BRU 85].

From the obtained association graph it is interesting to extract informations which are more easily interpretable at a semantic level. The solution we chosed is to extract cliques, or maximal complete sub-graphs after eliminating the weakest liaisons.

Because we decided to concentrate on noun groups as basic information chunks in the indexing process, verbs are not considered in the association graph and thus in the extracted cliques.

From experimental results we note that cliques are similar to noun phrases, and that words in large cliques occur in same sentence of the original text .

By construction cliques represent sets of tightly related words and may in turn be considered as representations of concept classes. Consider for example the clique <paper document on>. This information means that there is a complete contextual association (here a triangle) among these three words. The substantives give semantic information by themselves, and this is also true for "paper document", "document on paper". Thus the considered clique constitutes a set of concepts some being relevant, some other not (like "paper on document") because the clique's component are not ordered.

For storage efficiency cliques differing only from connective words such as prepositions may be combined in a single one. Then the clique <document paper of> may be combined to the former one giving the unique <paper document on of>. Moreover, cliques give a definition of the semantic context of any word (it should be noticed that a given word, or subset of words, may belong to several distinct cliques).

2.2. The indexing process :

We have defined and experimented an automatic indexing method for natural language texts [KER 84], [KER 85], which uses the above described partial knowledge model to normalize the indexing terms [BRU 85], [CKB 86].

This method is based on the exploitation of structural properties of texts :

- the logical structure of the texts allows the definition of a dynamic indexing process,
- the structural properties of natural language allows the definition of the indexing elements,
- this strategy is also based on a set of statistic criteria.

2.2.1. Exploitation of the logical structure :

The logical structure of the document describes the hierarchical organization of the informations (for example chapter, subchapter, paragraph, ...). In the following text we call **structural entity** the elements which correspond to any sub-tree of the hierarchical structure. In the considered strategy, the corpus is thus considered as a file of structural entities. On the other hand particular structural indicators such as titles often give important semantic informations which can be used to index the document.

2.2.2. Indexing term notion :

The notion of indexing terms is closely related to noun phrases; the basis of the indexing process is thus the extracting of such groups from sentences, according to the following syntax :

CONCEPT ::= S1 [ADJECTIVE]* [PREP S2]^2
S1 ::= NOUN / NOUN NOUN
S2 ::= S1 [ADJECTIVE]*

These groups are normalized via a confrontation with the thesaurus items. This is done through a pattern matching procedure between the concepts, produced by the syntactic process, and those stored in the knowledge base.

Moreover the pattern-matching procedure controls, through normalization any excessive growth of the indexing term set. At the end of this pattern-matching operation, each element of the structure is associated with a set of indexing terms which are then stored in the indexing base.

2.2.3. Statistical criteria :

The indexing relation is weighted into [0,1] by a statistical criteria rather than a selective strategy, based on statistic elements which are the concept frequencies in the different indexing units [RIJ 79], [SAL 83]. Aside these local measures, the statistic criteria are also related to the global behavior of indexing terms in the corpus.

These local and global properties of the criteria are respectively expressed by the notions of "concept representativity within a document content", and "document representativity of a concept within the whole corpus" [KER 84].

2.2.4. Dynamic indexing :

The dynamic indexing strategy is based on document hierarchical aspects and on some indicators such as the title of structural entities. The indexing of the different units is performed in an ascending order; we first analyze the minimal level indexing units and we recursively apply the previously defined strategy to the maximal level indexing units.

3. EXPERIMENTS :

Experiments are based on a test corpus extracted from a technical document : the NEF which is a reference manual used by engineers involved in autocommutation engineering). This text is highly structured and is organized in 15 main chapters, each of which deeply subdivized in sections, sub-sections ... The size may be evaluated around 30 000 words belonging to a dictionary of 4415 different forms. The corresponding lemmatized dictionary contains 2643 lemmas.
For testing the query system we considered only one of the largest chapter whose size is 2853 words. Inflected and lemmatized dictionaries contain 2853 and 1064 elements (roughly 60% and 40% of the global dictionaries). The knowledge base currently contains 809 cliques. It is interesting to note that only 400 (about 40%) of the lemmas occur in the cliques; this means that up to 60% of the words do not participate to significant concepts. The indexing base contains 467 indexing terms. Comparison with manual indexing through noun groups shows that the system was able to find up to 80% of the relevant indexing terms; the remaining 20% not being found were due to pronominal references and failures in the morphological analysis (ill recognized grammatical categories which misled analysis process).

Except for the text itself, all the data is stored according to LISP representation, with an extensive use of LISP properties which are a convenient way to represent associations (see figure 1) between sets. Furthermore LISP ensures good optimization of space and provides quick access to data. On

the other hand it lacks (at the moment) efficient functions for external memory management so we are restricted to small corpus until improved versions are available.

The figure below summarizes the sets and associations connecting them which constitute the core of the system. Sets are named and associations are figured by named arrows. An upper case initial for an association name denotes a 1 : n (one to many) association, a lower case initial denotes a 1 : 1 (one to one) association. As an example, the association "lemma" is one to one which means that for every x belonging to the inflected dictionary, lemma (x) is reduced to one element (its lemma).

(1) inflected dictionary —lemma→ lemmatized dictionary (Inflexion)
= {inflected forms} = { lemmas}

(2) lemmatized dictionary —Clique→ knowledge base (Contains)
= { lemmas} = { cliques}

(3) knowledge base —Indexing-terms→ Indexing base
= {cliques} = { indexing terms}

(4) Indexing base —Refers-to→ texts reference
x real ∈ [0,1]
= { indexing terms} = { (text-reference, p∈[0,1])}

(5) text reference —refers-to→ text fragments
= { text references } = { full text fragment}

The association **Clique** is multivalued which means that for every x of the lemmatized dictionary **Clique(x)** may contain more than one element (i.e.there are several cliques containing x as an element). Line (4) introduces the indexing relation, which is a weighted association between an indexing term and a set of textual references (i.e symbolic addresses in the logical structure of the document).

4. THE QUERY SUBSYSTEM

4.1. Overall presentation

The query subsystem is designed to fulfill the following goals :

- a friendly man-machine communication, based on a natural language interface and hierarchical maps. Other experiments are also on their way toward graphic interfaces involving a window manager, a bit-map screen and a mouse.
- enhanced intelligent capabilities of the system, and thus a better query interpretation. This is currently based on conceptual pattern matching between query concepts and the knowledge base.

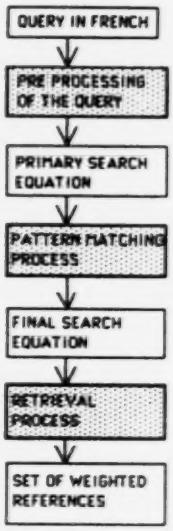
As mentioned above, the system handles structured documents. This allows great improvements in the answer formulation : references obtained are those of text fragments. Moreover, these references are sorted according to a relevance measure.

4.2. The software components

The query subsystem is made of several modules each of which being designed for a specific function :

- a query pre-processor which extracts concepts (i.e noun phrases) from the query and searches for logical operators connecting them. This implies linguistic tools such as morphological analysis and syntactical parsers.
- a pattern matching module which processes inferences from query concepts toward system known concepts (i.e indexing terms).
- a retrieval module which finds the relevant set of references using a Final Search Equation (FSE) obtained from the original query.

Of course this set of modules has to interact while processing a query; they are sequentially related as shown in figure 2.



- FIGURE 2 -

4.3. The query pre-processor

The query pre-processor analyzes queries and produces a primary search equation (PSE), which is a boolean expression of concepts. Let us first define the query language, and then describe the way we obtain corresponding PSEs.

4.3.1. The query language

The query language is the French language, but its interpretation is restricted to what may be recognized as a boolean expression of French noun phrases. The system always display its own understanding of the query, so the user can restrict himself to less ambiguous formulations when it becomes obvious that the system was mislead by an intricate formulation of the query (such as embedded relative subordinates and other well known niceties of

natural languages). The only restriction is that every word must be known by the system; any query is thus viewed as a succession of sentence fragments eventually separated by logical operators (conjunctions) :

QUERY ::= WORD$^+$ [OP WORD$^+$]*
WORD ::= items belonging to the system's dictionary
OP ::= AND / OR / EXCEPT

This is clearly not related to deep understanding of sentences [EIS 85], [SCH 75], [SCH 80], but rather close to heuristic pattern matching techniques already experimented in earlier intelligent systems [BAT 83], [BOG 82], [GUI 83]. We are of course interested by deep understanding and a first prototype was developed last year [NIE 85] in PROLOG, and was able to distinguish, for instance, external search attributes from content search attributes such as in the following queries :

"I want the books about Mao Zedong"
"I want the books from Mao Zedong"

By the way, the parser also identifies the external attribute "book" as a selection criteria among all the document types (i.e only the book documents will be selected).

This more evolved tool will be integrated in the next version of IOTA, as the PSE generator.

4.3.2. The PSE definition

The Primary Search Equation is the first step in the query understanding; it is basically a boolean expression of noun phrases extracted from the original query. PSEs may be syntactically defined as follows :

PSE ::= CONCEPT [OP PSE]*
OP ::= AND / OR / EXCEPT
CONCEPT ::= NOUNGROUP (see below)

Transforming a query into its PSE representation implies a morpho-syntactical analysis :

- the sentence is partitioned in fragments around logical operators the arity of which is controlled.
- concepts are extracted from the above fragments. Alike while indexing, our main assumption is that important search concepts lie within noun-groups. When several distinct concepts are extracted from a unique query fragment, they are considered as being related by the AND operator.

The syntax of noun-groups is the following :

NOUNGROUP ::= SIMPLENOUN [PREP SIMPLENOUN]^2
SIMPLENOUN ::= CNOUN [ADJ]* / [ADJ]* CNOUN
CNOUN ::= NOUN / NOUN NOUN
ADJ ::= adjectives
NOUN ::= nouns
PREP ::= prepositions

The parser searches within a query fragment for a concept's beginning (i.e adjective or noun) and then applies a transition network to recognize the whole noun groups.

Noun-groups are then reorganized (a first step toward normalization) : adjectives related to a given noun (if any) are sorted in alphabetic order, and this list is appended to the noun.

Consider the query :
"I want everything concerning distributed data-bases and management"
the corresponding PSE will be :
(data-base distributed) AND (management)

4.4. The pattern matching process

Given a PSE, the problem is now to consider whether primary concepts extracted from the query fit with the indexing terms or not. If they do, the PSE may be processed by the retrieval module; if they do not we need further transformations to adjust external knowledge to internal knowledge. We consider two steps in this process :

- the transformation of external concepts into thesaurus concepts, the thesaurus is here used as a fuzzy index to the indexing terms.
- access to the indexing terms.

a) The transformation step :
Given an external concept, we perform a pattern matching on the thesaurus, made of the following operations :

a-1) inclusion step :
We look for a thesaurus item (i.e a clique) which includes the given group. If there is one, go to step b "access to the indexing relation", otherwise go to step a-2 below.

a-2) transformation step :
If the original group does not fit any thesaurus item, there is no chance that we can find an indexing terms which can fit in. Then we break the original group in sub-groups, which are in turn submitted to step a-1. The breaking process stops when every sub-groups satisfies a-1 (possibly isolated words), and is based on syntactical rules to avoid the generation of syntactically inconsistent sub-groups. The strategy here is to limitate the loss of information while breaking the noun group. Thus adjectives (if any) are removed first. If this is not enough the group is broken around prepositions and so on.
IOTA uses a syntactical net to control this operation, made of breaking rules for every syntactical pattern of noun groups (as they are defined in 4.3.2 above). When step a-2 is achieved, the process goes on to step b.

b) Access to the indexing relation :
As previously described, each thesaurus item is related to a set of indexing terms (i.e noun groups which are ordered subsets of this item). In this search step, the matched query term is compared with a definite indexing term list. Two cases may occur :

b-1) the group exactly matches an indexing term :
The process ends for this group by replacing it in the PSE by the corresponding weighted list of references.

b-2) the group does not match any indexing term :
This means that the group, though a consistent concept, was not registered as an indexing term. Since there is no semantic relationship available on this early version, we can only look for syntactically related concepts in this list (semantic relationship will be managed by the expert system we are developing). Syntactically related concepts means, in this case, specifics : we consider that adding a qualification to a concept leads to a more specific concept, like "distributed data-base" is a specific of "data-base". Thus if we have to process "data-base" and if there is no such indexing term, we look for the union of all the syntactical specifics of this term, and consider the resulting list of references as the result for this term (i.e the union of these reference lists is substituted to the original term in the PSE). If there is no specific associated to the term, we activate again the breaking process described in a-2.

At the end of steps a) and b), the original PSE is transformed into a boolean expression of reference lists. Note that additional logical operators may have been added when original groups have been broken during step a-2; a group such as "distributed data-base in management", if broken in "distributed data-base" and "management", will lead to the following sub-expression :

AND(L("data-base distributed"),L("management"))

where L(x) stands for the list of references associated to the indexing term x..

We call this final state of the search equation the Final Search Equation (FSE). Note also that the pattern matching process always ends, at least with a FSE involving only lists of isolated words as indexing terms (cf our hypothesis : every word of the query is known by the system).

4.5. The retrieval process

A FSE is basically a logical expression whose operands are lists of weighted references (subsets of the indexing relation); the answer is thus the subset of weighted references satisfying this boolean expression. Because references are subtrees, there is necessarily a special interpretation of the operators AND, OR and EXCEPT due to the inclusion relation which may exist between two subtrees. We first describe the algorithms related to the logical operators in this context, and then give the definition of the weighting functions associated to them.

a) Reference conjunction : AND
Let L1 and L2 be reference lists; AND(L1,L2) corresponds to the list of subtrees which belong to both L1 and L2. Such subtrees are found according to the following algorithm (a_1 is an element of L1, a_2 an element of L2) :

- $a_1 = a_2$: a_1 belongs to the result,
- a_1 is a subtree of a_2 : a_1 belongs to the result,
- a_1 and a_2 are disjoint trees : neither a_1 nor a_2 belongs to the result.

b) Reference disjunction : OR
OR(L1,L2) corresponds to the list of subtrees belonging either to L1 or L2 or both. The OR algorithm is the following, where a_1 and a_2 have the same meaning as above :

- $a_1 = a_2$: a_1 belongs to the result,
- a_1 is a subtree of a_2 : a_2 belongs to the result,
- a_1 and a_2 are distinct subtrees : a_1 and a_2 belongs to the result.

c) Reference negation : EXCEPT
EXCEPT(L1,L2) corresponds to the sublist of L1 whose references do not include any element of L2; the elimination process is the following, considering an element a_2 of L2 :

- $a_1 = a_2$: a_1 is discarded from the result list,
- a_1 is a subtree of a_2 : a_1 is discarded from the result list,
- a_2 is a subtree of a_1 : a_2 is pruned from a_1, the result is the set of highest level subtrees of a_1 which do not contain a_2 (pruning of a_1),
- a_1 and a_2 are disjoint : a_1 belongs to the result.

In order to evaluate relevance of result references, weighting functions are associated to the above logical operators. They all use the basic weights of the indexing relation, which were themselves based on relevance evaluation (see [KER 84] and [KER 85]), and are real functions ranging within [0,1].

The basic principle of this evaluation is to consider, given two references R1 and R2 with weights p_1 and p_2, a function Fop(p_1,p_2) whose value may be considered as a relevance measure to op(R1,R2). Such a function clearly depends on the operator 'op' and the weights p_1 and p_2 related to R1 and R2. Many definitions may be chosen for these functions, based on fuzzy set theory [ZAD 79], multivaluated logics [LUK 30], ...In this first version of IOTA, we experimented the following, which delivered good results at a reasonable cost :

a) Evaluation function for the disjunction operator : For
Because of the commutativity and associativity of the OR operator, the function has to have these properties :

$$For(R1,R2) = (p_1+p_2-p_1 \cdot p_2) / (1+p_1+p_2-2 \cdot p_1 \cdot p_2)$$

b) Evaluation function for the conjunction operator AND : Fand
The function must have the same properties as above :

$$Fand(R1,R2) = (p_1 \cdot p_2) / (1 - p_1 - p_2 + 2 \cdot p_1 \cdot p_2)$$

Note that these two functions are not idempotent, ie Fop(p,p) ≠ p. This means for example that the relevance measure for AND(R1,R1) is different from p_1. This is of course a bias at a theoretical level; in fact, the case is relevant only when the two occurrences of R1 are related to the same concept say C1. This means that we are solving a query such as AND(C1,C1) or OR(C1,C1) which clearly are rather dumb.
It can be shown anyway that :

Fand : if $p < 0,5$ Fand(p,p) < p
if $p > 0,5$ Fand(p,p) > p

For : if $p < 0,5$ For(p,p) > p
if $p > 0,5$ For(p,p) < p

c) Evaluation function for the except operator : Fexcept
This operation is not commutative, and may be considered as a AND(R1,NOT(R2)), hence the following definition derived from the Fand function :

$$Fexcept = Fand(p_1,(1 - p_2))$$

Of course, idempotence does not have sense for Fexcept since EXCEPT(A,A) is the empty set.

4.6. Utilities and user helps

Some basic commands have been provided to help the user formulating queries; most of them are related to find accurate indexing terms. Recall that the system displays the final search equation (i.e its ultimate understanding of the query), so the user may choose to reformulate the query if he estimates this interpretation as not satisfactory.

a) Dictionaries :

The user, if he has troubles with words, may consult both original and lemmatized dictionaries. The first one displays, given a lemma, all the known inflexions of a word, the second displays lemmas from inflected words.

b) Cliques :

Recall that cliques are basic elements of the knowledge

base, and represent classes of system known concepts. The user thus can browse in the thesaurus in two ways, trying to refine query concepts :

- selecting cliques from a list of words : the system displays all the cliques containing this set of words,
- selecting sets of cliques using a pattern filter : a filter is a list of symbols which defines a clique pattern. The system will display every clique matching this pattern. Authorized symbols to define a filter are :

  - a lemma,
  - a syntactical category,
  - the symbol $ which stands for any simple lemma,
  - the symbol * which stands for any list of lemma,
  - the symbols (and) which introduce alternative searches.

  Filter access is very useful when searching what the system knows about a concept; the following example illustrates these capabilities :

  Examples :
  data-base * --> all the cliques related to "data-base"
  data-base $ --> all the 2-word cliques containing "data-base".
  data-base a --> all the 2-word cliques containing "data-base" and an adjective
  data-base (relational network) --> all the 2-word cliques containing either "data-base" "relational", or "data-base" "network"

  Of course, more complex patterns may be defined using these basic possibilities; the filter syntax is the following :

FILTER ::= ELEMENT$^+$
ELEMENT::= LEMMA/CATEGORY/$/*/ALTERNATIVE
ALTERNATIVE ::= ([LEMMA / CATEGORY]$^+$)
LEMMA ::= element of the lemmatized dictionary
CATEGORY ::= morphological category

c) Indexing terms :

Given a set of words we display all the cliques containing this set and the indexing terms associated to them.

d) Text units :

Given their reference any text fragment may be displayed, along with its title and the set of indexing terms related to it.

5. CONCLUSION

Although incomplete, and tested on small corpus, IOTA has demonstrated very interesting possibilities, critical steps such as query pre-processing and pattern-matching are efficient and give good results. The system is currently implemented on a VAX 11/780 under UNIX 4.2 BSD system, and is written in LE_LISP.

It is obvious that a critical point lies in the size of the corpus such a system can manage, and in particular the respective growth of the corpus and the thesaurus, along with the indexing relation. Further experiments are planned to load the whole document mentioned in chapter 3 (up to 100.000 words) but this implies serious adjustments towards text storage and access techniques. On the other hand knowledge storage may be still improved by compaction techniques applied to cliques and indexing terms.

In the next future the main improvement will be provided by the integration of the query expert system we are developing and which is almost achieved. This major component will act as the supervisor of all the previously defined modules, with additional capabilities such as automatic reformulation of queries and the involvement of classical semantic relations when generating the Final Search Equation.

Another significant advance will be the integration of more sophisticated linguistic processors (a morpho-syntactic parser) which will greatly improve communication flexibility and save a lot of lexical data (we will need only the lemmatized dictionary).

Thus clearly IOTA is a first important step in our study as the backbone of further developments and a tool for experiments on various corpus.

6. REFERENCES :

[BAT 83] M BATES, R.J BOBROW :
"Information retrieval using a transportable natural language interface"
ACM SIGIR, 1983

[BOG 82] B.K BOGURAEV, K SPARCK JONES :
"A natural language analyzer for database access"
Information Technology : Research and Development 1982, 1 (23-39)

[BRU 85] M.F.BRUANDET :
"Modèle partiel de connaissance pour un système de recherche d'informations"
RIAO 85 18-20 mars 1985 GRENOBLE.

[CKB 86] Y.CHIARAMELLA, D.KERKOUBA, M.F.BRUANDET :
"Intégration d'une fonction documentaire dans un atelier de logiciel"
Journées CONCERTO, Perros Guirrec, France, feb 1986 (to appear)

[DEF 85] B.DEFUDE :
"Different levels of expertise for an expert system in Information Retrieval"
8th ACM SIGIR Conference, Montreal, Canada, june 1985

[EIS 85] K.P EISELT :
"A parallel-Process Model of On-Line Inference Processing"
in Proc. IJCAI-85

[GUI 83] G GUIDA, C TASSO :
"IR-NLI : an expert natural language interface to online data bases"
Proc. Conf. on Applied Natural Language Processing, Santa Monica, 1983

[KER 84] D.KERKOUBA :
"Une méthode d'indexation automatique des documents fondée sur l'exploitation de leurs propriétés structurelles. Application à un corpus technique"
thèse 3ième cycle INPG 1984.

[KER 85] D.KERKOUBA :
"Indexation automatique et aspects structurels du texte"
RIAO 85 18-20 mars 1985 GRENOBLE.

[LUK 30] J.LUKASIEWICZ :
"Many-valued systems of propositional logic"
in S.McCall, Polish Logic, O.U.P, 1957

[NIE 85] J.NIE :
"Compréhension de requêtes en langue naturelle. Pré-étude dans le contexte d'un système de recherche d'informations"
Rapport de DEA INPG/USMG, Grenoble, 1985

[RIJ 79] C.J.VAN RIJSBERGEN :
"Information retrieval"
Second édition, Buttersworth London England 1979.

[SAL 83] G.SALTON, M.J.McGILL :
"Introduction to modern information retrieval"
Mcgraw Hill Book Company, New York, 1983 .

[SCH 75] R.C SCHANK :
"Conceptual information processing"
Amsterdam, North Holland, 1975

[SCH 80] R.C SCHANK, M LEBOWITZ, L BIRNBAUM :
"An integrated understander"
in American Journal of Computational Linguistics, vol 6, N 1, 1980

[ZAD 79] L.A ZADEH :
"Approximate Reasoning Based on Fuzzy Logic"
in Proc. IJCAI-79, Tokyo, August, 1979, pp 1004-1010

AN INFORMATION RETRIEVAL SYSTEM BASED ON ARTIFICIAL INTELLIGENCE TECHNIQUES

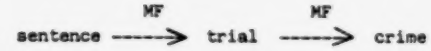
Dario De Jaco, Gianluca Garbolino
Laboratorio di Intelligenza Artificiale
CSI-Piemonte - C.so U. Sovietica, 216 - 10134 TORINO (ITALY)

ABSTRACT

This paper describes a possible use of Artificial Intelligence models and techniques in the design of a small Information Retrieval system. In particular, some knowledge representation models, such as semantic networks and frame-like structures, are viewed as interesting tools for the implementation of a thesaurus, and also for a description of the stored documents' contents. In addition, a parser based on the ATN (Augmented Transition Network) model which can analyze Italian sentences concerning a legal domain is described. We are including it in an user/system interface whose goal is to provide the user with the possibility of expressing search topics by using noun phrases or other linguistic expressions, rather than single words or Boolean combinations of them. Finally, some tasks requiring automated reasoning facilities are outlined.
The kernel of the system, i.e. the component which both performs traditional information retrieval and allows the insertion of new documents, is described in an appendix. It was first applied to a bibliographic database containing about a thousand references (with abstracts) of both papers and books concerned with Artificial Intelligence; now we are working on its application to a legal domain, with a database of laws, decrees and sentences conerning pollution and environmental protection.

1. Knowledge representation

We believe an Information Retrieval system should be provided with a model of the domain under consideration, i.e. with some representation of the relevant knowledge. Many systems use thesauri which, from this point of view, store some knowledge about the terms they include. In fact, they often record a few hierarchical relations among the terms themselves. For this reason, we would like to think of a thesaurus as a network rather than as a simple list of words. Thus semantic networks appear to be particularly adequate as a knowledge representation model (SIMM73), (HEND76). Each term in the thesaurus will be represented by a node in the corresponding semantic network, while each relation will be represented by an arc. Since semantic networks have never been defined in an uniform way but they have often been employed in computer systems having very different goals, there is in principle no restriction to the variety of arcs one can use to connect nodes (1). As a consequence, semantic networks can represent various classes of relations among words, including both the hierarchical relations of the traditional thesauri (e.g. BT, NT, etc.) and other kinds of relations such as temporal, causal and others. For example, within a legal domain, we may whish to say that a sentence may follow (in time) a trial, which, in turn, may follow a crime. By introducing a MF (May Follow) arc, we can represent such knowledge by the following semantic network's fragment:



It is clear that most of the relations one can define do not generally hold, but depend closely on the domain under consideration so that they can be defined only after the domain has been chosen. However, we think that defining a significant number of such relations would increase the expressive power of a thesaurus, and would consequently improve the retrieval of documents.

---

(1) From a computational point of view this is not completely true, in the sense the difficulty of management grows with the complexity of the model.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.
© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

Another extension that comes from the adoption of semantic networks as a model for the design of thesauri is related to the nodes. In fact, a node may represent not only objects that are denoted by single words, but also situations that are denoted by more complex linguistic expressions, such as noun phrases. In order to represent an expression of this kind, we think of a sort of structured node, in which the expression is represented both in terms of the words composing it, and in terms of the relations connecting them. As an example in the usual legal domain, suppose you want to build a node for the noun phrase 'defence of health'. Such a node may be represented in the following LISP-like notation:

(defence (OBJ (health))

Unlike a simpler representation in the form of a string, this one indicates the kind of relation (in this case a sort of object-relation) between the head of the noun phrase and the remaining component. From the point of view of the representation, structured nodes may be viewed as frame-like structures. Thus we obtain two levels of knowledge representation: a more general level, emphasizing semantic relations among terms or concepts, and a more specific level, in which concepts are defined in terms of component words and relations between them. At the former level we find a semantic network whose nodes and arcs are thesaurus terms and relations respectively, while at the latter there are both simple symbols and frame-like structures representing single terms and concepts respectively.

Another kind of knowledge an Information Retrieval system should be provided with concerns the content of the stored documents. To this purpose, many systems include lists of indexes. As is well known, they consist of either single words or groups of words which were previously assigned to the documents in a manual, semiautomatic, or fully automatic way. We think that a more detailed description of a document could consist of a list of the most important concepts occurring in the document. As in the case of the thesaurus, if such concepts are denoted by single words their representations would be the words themselves, while if they are denoted by noun phrases, their representations would be frame-like structures. We think such representations of the describing phrases assigned to the documents would refine the retrieval of the documents themselves, to the detriment of computational efficiency.

2. Natural language analysis

As is well known, a particular field of Artificial Intelligence is concerned with natural language processing. Within this field, systems have been implemented that "can understand" small fragments of language in well-defined domains. Although the problem is still completely open, there are techniques of syntactic-semantic analysis of language that appear to be applicable in Information Retrieval. In particular, we foresee two areas of application. The first concerns the insertion of the documents. Here some program able to control the phrases assigned to the documents as indexes would be very useful. Such a program should (e.g.) present the user with a set of particular linguistic expressions, such as noun phrases, previously extracted from the texts by means of syntactic analysis. Further, if we suppose the above phrases to express the most important concepts contained in a document, a more powerful program should then build frame-like structures starting from them. Such a program, which should include a lot of semantics, is a final goal of our research.

The other area of application concerns the retrieval of documents. Here the goal is to build a user/system interface which would allow the user to express search topics rather than single words or Boolean combination of them. The inputs should be represented according to the chosen representation model, in order to match them with the knowledge base and to retrieve documents. .

We already implemented a parser based on the ATN (Augmented Transition Network) model introduced by Woods in 1970 (WOOD70); now we are working on its inclusion both in the interface and in the program which extracts noun phrases from texts.

2.1 The parser

It has been implemented in two different versions: the first is a Franz LISP version running on VAX-11/780, whereas the second is a GCLISP version for M24 personal computers. Such a parser, which can analyze Italian sentences (one at a time) concerning a legal domain, uses for this task a lexicon of about 350 words and a grammar weakly equivalent to a context-free grammar (WINO83). Although it is based on ATNs, it differs from this model for the following reason especially. An ATN should be able to build every possible syntactic structure of a given sentence. On the contrary, our parser assigns a sentence one representation only. In order to build such a representation it may use some semantic information stored in the lexicon. The major advantage of our approach consists in reducing computation's time (because there are no heavy backtracking operations), whereas the obvious shortcoming is that if the parser has built a wrong structure, there is no way of modifying it.

The lexicon has been implemented in LISP by means of property lists. Each entry of the lexicon, which can be either a lemma or a derived form, is given a list of ⟨attribute, value⟩ pairs. In the case of lemmas, both syntactic and semantic features are included in the list. Features of the former kind determine the grammatical category of a given word, its gender, and its number. Semantic features define logical relations between concepts (e.g. membership, subset, superset, etc.). Moreover, a sort of case structure (BRUC75) has been added to every entry corresponding to the infinitive of a verb (the lemma), in order to account for the acceptability of the sentences in which that verb occurs. This is done in terms of selection restrictions which place semantic constraints on the objects which fill the case slots. For example, the case structure associated with the verb emettere is:

```
(emettere (AGENT organo-giudiziario)
          (OBJECT provvedimento))
```

meaning that a judicial body (organo giudiziario) can issue (emettere) an order (provvedimento) but the reverse (e.g.) cannot be said.
In the case of derived forms, only those features that distinguish them from their lemmas have been included in the corresponding entries, whereas common features are inherited from the lemmas themselves.

The parser uses semantic information in order to control prepositions and verbs. In the first case the task consists in determining the correct position of a prepositional phrase (PP) within a sentence. As an example, let us consider the following noun phrase:

sentenze di condanna della Corte

where the problem is to interpret della Corte as modifying sentenze rather than condanna. In order to deal with problems like this, we provided each simple preposition stored in the lexicon with a list of ordered pairs of words, with the property that, for each pair belonging to the list, we can assign a meaning to the prepositional phrase obtained by putting the given preposition between the words of that pair. Moreover, this property extends to any unprovided pair of words which express more specific concepts than those expressed by a provided one (2).

Going back to our example, since the parser works top-down and depth-first, it first tries to couple della Corte with condanna. For this purpose, it searches the list assigned to the preposition di within the lexicon either for the pair:

(condanna Corte)

itself, or for any pair (X Y) where X and Y express more general concepts than condanna and Corte respectively. But, since it does not find any such pair, it fails. So the parser closes the noun phrase condanna, builds the prepositional phrase di condanna, takes again into account the noun phrase starting with sentenze (here the semantic check will succeed); finally, it builds the following structure:

```
(NP (nc sentenze)
    (PP (prep di)
        (NP (nc condanna)))
    (PP (prep della)
        (NP (nc Corte))))
```

With regard to verbs, we have taken into account the task of determining the role of the pronoun che within possible relative clauses. In order to achieve this goal, the parser performs a consistency test between words from the input sentence and information from the case structure associated with the verb of the relative clause. Consider for example the following noun phrase:

sentenza che ha emesso la Corte di Torino

In this case the parser can establish that che is the object of the relative clause (it was the Court who issued the sentence) since:

1) the case structure of the verb emettere states that the agent must be an organo giudiziario while the object must be some kind of provvedimento;

2) the words Corte and sentenza are related through 'subset' links to organo giudiziario and provvedimento respectively.

Thus the syntactic structure resulting from the analysis of the whole noun phrase is:

```
(NP (nc sentenza)
    (RC (NP (det la)
            (nc Corte)
            (PP (prep di)
                (NP (npr Torino))))
        (VP (v ha emesso)
            (NP (pron-rel che)))))
```

Disregarding readability, this structure should remarkably facilitate a subsequent program which would have to build from it (or from other structures of the same form) a representation

---

(2) Let us be reminded that words in lexicon are linked together by relations such as 'subset', 'superset', etc.

according to the frame-like model. In fact, such a program should always assign the roles of subject and object to NP1 and NP2 respectively, within contexts of the form:

NP1-VP, where VP=v-NP2.

2.2. User/system interface

As it has been said, the major goal of the user/system interface is to provide the user with the possibility of expressing search topics by using linguistic expressions more complex than single words or groups of words. In fact, it is clear that, even in a specific, well-defined context, the aggregation of words by means of Boolean operators is inadequate in many situations. As an example, suppose you have a bibliographic database containing documents dealing with Computer Science; suppose further you would like to retrieve documents relevant to the following different concepts:

1) interrogazione di sistemi
   (system consultation)

2) sistemi di interrogazione
   (consultation systems)

In order to deal with such situations, some Information Retrieval systems provide the users with operators (e.g. ADJ) allowing them to define the sequence in which two given words must be found. Such operators may also allow users to specify how many words can separate the two given words. There tends to be something artificial in such solutions: the trick may work well in some cases, but not in many others.

Artificial Intelligence suggests a very different approach, which takes into account the meaning of the words, and consequently the meaning of compound linguistic expressions. As it has been said in the first section of this paper, if we can represent both noun phrases in the above example by means of frame-like structures such as:

1) (interrogazione (OBJ sistemi))

2) (sistemi (FOR interrogazione))

then there is a way of distinguishing the one from the other that appears to be more powerful than the simple word-order. Such structures are to be matched with similar structures that address the documents from the knowledge base; such a match is a complex one, because it must take into account not only the words but also the relation between them.

2.3. Text analysis

At the beginning of section 1.2 we outlined two different levels of text analysis: a first level, mainly concerned with syntax, whose goal is to partially support the task of assigning descriptive phrases to the documents, and a second, deeper level, whose goal is to build internal representations of such phrases. Now we have a program which can perform text analysis at the first level. Such a program, which includes a version of the described parser, can extract noun phrases starting from a text. Its goal is to help the user to select words and phrases as indexes to that text. Our intention is to develop this program in order to achieve a level of analysis increasingly closer to the second level. Here the goals are: first, to select among the noun phrases those expressing the most important concepts contained in a text, and second, to represent them according to the frame-like structure and store them in the knowledge base.

There is a third level of analysis we would like to mention here, although it seems to be closer to the research on text understanding in Artificial Intelligence rather than to the aim of text analysis in Information Retrieval. At this level a text is viewed in its entirety as composed by variously interrelated concepts rather than as a list of separate ones. The relations among the concepts describing a text should be either defined or clarified by a program which could match the concepts under consideration against the knowledge base, where, as we said in the first section of the paper, these relations have been previously represented (ALTE85).

3. Automated reasoning

Another important field of Artificial Intelligence we are interested in concerns automated reasoning. In particular, we would like to provide the system with an inference mechanism allowing it to:

1) retrieve information not explicitly stated in the database, but such as can be inferred by means of other information stored in it;

2) retrieve information logically connected to the user's explicity request.

As an example of what we mean, let us mention RESEDA (ZARR84), an Information Retrieval system acting on a factual database that contains some biographies of French Middle Age personages. This system, employing Artificial Intelligence techniques, is able to point out the hidden causes of an event stored in the database, using a set of inference rules, the "hipotheses", that show causal

relations among classes of events. In some detail, when a certain fact is found in the database as a direct answer to a specific user's request, the system selects any rule causally connecting this fact to other classes of events, through a pattern-matching between the fact and the antecedent parts of the rules; then the system employs the consequent parts of the selected rules in order to generate new search plans to find out facts compatible with them.

4. Conclusion

Techinques of natural-language analysis currently studied in Information Retrieval often leave out of consideration linguistic meaning. Therefore, many systems are not so helpful to the users, failing to meet their needs. An Artificial-Intelligence-based approach would rather think of systems having some skills of natural-language understanding. Systems of this kind should be able to select documents by matching conceptual representations rather than strings of characters. Nevertheless, their application is bounded to well-defined domains, where all the knowledge necessary for the systems to run could be represented and managed in a satisfactory way.

APPENDIX

The system's kernel

This program was conceived as the core of a more complex system. However, it can run by itself as a simple Information Retrieval system. It has two major components:

1) FINDER which enables on-line retrieval operations,
2) BUILDER which enables the insertion of new documents,

and it is implemented in two different versions, each requiring a particular hardware configuration. The first one is entirely written in Golden Common LISP (GCLISP), and runs on Olivetti M24 personal computers (MS-DOS operating system) with main memory and hard disk of 640 K-bytes and 10 M-bytes respectively. This version is mainly addressed to a single user; as it can only manage small databases, it has a mainly experimental character. However, it is possible to use it for practical purposes: for instance, it could be applied to a collection of documents concerned with a particular historical period, as stored in a city's State archives (1).

---

(1) One of the experiences we made in the past few years was the construction of a database containing all the Bulls issued by anti-pope Felix V between 1440 and 1448.

The second version needs a more complex hardware configuration, including the following components:

1) one AT&T 3B2/300 mini-computer (UNIX operating system, up to 75 M-bytes of hard disk space) as main processor also storing the database;

2) up to four Olivetti M24 as terminals with local processing capabilities.

This configuration, which can be realized by means of PC-Interface, a software tool which enables the MS-DOS and UNIX operating systems to communicate, sharply distinguishes between the system's user and the system's administrator. The former is only enabled to use the FINDER program, while the latter can use the BUILDER program as well. The former program is written in GCLISP, as in the previous version, and runs on M24, while the latter is now written in Franz LISP, and runs on 3B2/300.

1. The bibliographic database

This database does not contain whole documents but a kind of substitutes for documents: the bibliographic items. Each item describes a particular document by means of some attributes belonging to a predetermined set: the item's model. Let us consider two main subsets of this model:
- the least subset for document description
- the subset of search channels .

The meaning of the first, which is formed by the AUTHOR, AFFILIATION, TITLE, DATE, ABSTRACT, and KEYWORDS attributes, is that every item in the database must contain all its attributes, whereas the second, which includes AUTHOR, AFFILIATION, EDITOR, TITLE, DATE, and KEYWORDS defines the attributes where the system can search for particular values in order to retrieve items. Particularly, the value of the KEYWORDS attribute is, for each item, a list of controlled terms assigned to the correspondent document by the system's administrator. At present, the system is not allowed to search for terms in the abstracts, that being a field in which we would like to use Artificial Intelligence techniques.

Each item is stored in its own file on disk space. This way, the task of retrieving items is left to operating system's routines, whereas the Information Retrieval system only has to build their full path-names. The association of an item to a file is a task for the BUILDER program.

2. The FINDER program

It is composed of a set of LISP functions which allow the user to:

1) select collections of items by specifying particular values of attributes defined as search channels;
2) aggregate collections by using Boolean operators such as AND, OR and NOT;
3) select particular items from previously defined collections;
4) display and/or print whole collections, their parts, single items, ordered lists of authors, affiliations, titles, controlled terms;
5) store on disk the current search session (i.e. the whole history of the performed operations) before leaving it, in order to restore it in the future.

Users can directly call these functions, one at a time, in the GCLISP environment. For instance, the following S-expression triggers the $FIND function:

```
($FIND "logic-programming" 'KEYWORDS)
```

which will set up a collection of items including the "logic-programming" term as a value of the KEYWORDS attribute. Each collection is stored in the main memory as a LISP list of all the logic identifiers of files containing the related items. Moreover, the system assigns to each collection both a progressive integer enabling the user to identify the given collection, and a comment-list helping him to remember its meaning. That list is similar to the S-expression used to call the function setting up the collection.

Retrieval of items from the database is based on inverted files. They are automatically built at the moment of document insertion, and are associated with the search channels in one-to-one correspondence. Within the inverted files terms are looked up using a binary search algorithm. In order to reduce access time, the search operation is limited to that section of the inverted file which includes only terms starting with the same character as the term searched for. The sections' boundaries are defined in special "by-initial access tables", which are also automatically built and kept in the main memory.

AND, OR and NOT Boolean operations can build new collections from already existing ones. They correspond to the set-theoretical operations of intersection, union and complementation, respectively. They can also be nested in order to set up complex search strategies. These are executed by the LISP interpreter directly:

```
($AND ($OR 1 2)
      ($OR 3 ($FIND "Prolog" 'TITLE)))
```

Here we have assumed the numbers 1, 2, 3 to refer to previously constructed collections.

3. The BUILDER program

The goal of this program is to expand the database. Initially, the system's administrator compiles the new items and inserts them into a file. The BUILDER program then performs the following operations:

1) it writes any new item into a file onto one of the directories which were reserved for the database, and assigns a (user-hidden) symbolic name to the file;
2) it updates the inverted files;
3) it updates the by-initial tables that give acces to the inverted files.

Step (2) is the most complex; it is performed as follows. The system examines the new items one at a time, only taking into consideration the search channels subset of the attributes, and extracts from each of them a list of triples of the form:

<term,attribute,file-id>

where 'term' is any single word which is included in the value of the 'attribute' search channel, while 'file-id' is the logical name of the file which contains the item. The attribute TITLE requires separate mention. As in this case we have a sentence in natural language, but not a list of descriptors (as in the case of KEYWORDS), the system utilizes a short "negative dictionary", that allows it to disregard articles, prepositions, adverbs, etc. However, no kind of morphological word analysis is actually performed.

The lists that have been so determined are gathered in a single working file, the one that contained the lists of triples for the items currently in the database. From the working file other components of the BUILDER program build the final inverted files by way of splitting up, alphabetic merging, and packing.

References

(ALTE85) Alterman R., "A dictionary based on concept coherence", Artificial Intelligence 25 (1985), pp. 153-186.

(BRUC75) Bruce B., "Case systems for natural language", Artificial Intelligence 6 (1975), pp. 327-360.

(CROF85) Croft W.B., "An expert assistant for a document retrieval system", Proc. Conference RIAO, Grenoble (France), March 1985, pp. 131-149.

(FILL68) Fillmore C., "The case for case", in: Bach E. & Harms R. (Eds.), "Universals in linguistic theory", Holt, Rinehart & Winston, New York, 1968.

(FIND79) Findler N., "A heuristic information retrieval system based on associative networks", in: Findler N. (Ed.), "Associative networks", Academic Press, 1979.

(GUID83) Guida G., Tasso C., "An expert intermediary system for interactive document retrieval", Automatica 19 (1983), pp. 759-766.

(HEND76) Hendrix G.C., "Expanding the utility of semantic networks through partitioning", Artificial Intelligence 7 (1976), pp. 21-49.

(SALT83) Salton G., "A note on information retrieval models and theories", Proc. Conference RIAO, Grenoble (France) 1985, pp. 1-27.

(SALT85) Salton G., McGill M.J., "Introduction to modern information retrieval", McGraw-Hill 1983.

(SIMM72) Simmons R.F., Slocum J., "Generating English discourse from semantic networks", Comm. ACM 15 (1972), pp. 891-905.

(SIMM73) Simmons R.F., "Semantic networks: their computation and use for understanding English sentences", in Schank R.C., Colby K.M. (Eds.), "Computer models of thought and language", Freeman, 1973, pp. 63-113.

(WINO72) Winograd T., "Understanding natural language", Academic Press, 1972.

(WINO83) Winograd T., "Language as a cognitive process", Vol. 1: Syntax, Addison-Wesley, 1983.

(WOOD70) Woods W.A., "Transition network grammars for natural language analysis", Comm. ACM 13 (1970), pp. 591-606.

(WOOD77) Woods W.A., "Lunar rocks in natural English: explorations in natural language question answering", in: Zampolli A. (Ed.), "Linguistic structure processing", North Holland, 1977, pp. 521-569.

(ZARR84) Zarri G.P., "Expert systems and information retrieval: an experiment in the domain of biographical data management", Int. Journal Man-Machine Studies 20 (1984), pp. 87-100.

THE USE OF INFERENCE MECHANISMS TO IMPROVE THE RETRIEVAL FACILITIES FROM LARGE RELATIONAL DATABASES

Gian Piero Zarri

Centre National de la Recherche Scientifique
CNRS - LISH
54, boulevard Raspail
75270 Paris Cedex 06
France

ABSTRACT

This paper describes the development of "intelligent" tools aimed at improving the retrieval facilities from large relational databases. When a natural language query does not correspond directly to the data contained in the base, a class of inferential processes called "transformations" is applied. The original query is thus automatically converted into one or more "semantically close" ones. "Semantically close" means that the data possibly obtained with the new query will give useful information about the data originally searched for.

1. Introduction

The existing natural language interfaces to databases - too numerous to be cited, see for example <PYLY85> - free the user from learning the cumbersome and difficult *syntax* of formal database query languages. Still they impose a very *strict semantic model*, which corresponds to the way a real world domain is represented in a particular database.

They thus deal ineffectively with the absence of explicitly requested data.

For example, a query like "who knows anything about AI", sent to a staff management database, will normally get rejected, because this level of detail is not present in the database. The "cooperative systems" - <KAFL82> ; see also <FINI79>, <WEBB84>, etc. - get around this situation by informing the user that his assumptions about the content and structure of the database are wrong. However, this type of systems will not tell him that a *plausible answer* can be deduced from other related data present in the base.

The inference mechanisms proposed in this paper fall into this last area. Their aim is again to produce a "user oriented" and "friendly" behaviour (<WAHL84>, <WOOD84>, etc.). In our case, this is to be achieved *by searching semantic affinities between what is requested and what is present in the base* (<EUZE84a>, <EUZE84b>, <EUZE85>). For a related, more "logical" approach, concerning the problem of supporting the extraction of information implied by, but not explicitly included in, the contents of a relationally structured database, see <KELL78>.

The fundamental theoretical principle we follow is to transform the original query into one or more different queries, which are *semantically close*. This means that the new queries addressed to the base must be able to provide useful information in order to answer implicitly to the original query. Two new queries would be created in replacement of the request for experts in AI. If the company's Computer Division has an "AI" section, then a list of employees in this section would be given. If this is not possible or if a confirmation is needed, then a list of people working on AI projects will be created. Of course, the answers obtained in this way will not be 100% reliable, because they have been obtained by means of an inductive inference process. Once the transformation has been carried out, the system will ask the reliability and interest of the new queries before accessing the database. It should also be noted that the answers provided by this transformation process normally are a subset of the theoretically possible ones. In reality, there is no counter evidence that some other company's employees are qualified in AI, but happen not to satisfy the mentioned criteria.

A second very simple example of transformation concerns a hypothetical database about university professors. Suppose that you want to ask a question like "who has lived in the United States" even without an explicit representation of this fact in the database. If this base contains the degrees obtained by the professors together with the information where each one of them was born, we can tell the user that, although we do not explicitly know who has been in the U.S., we can nevertheless look for people born there and/or having an American degree. This last piece of information for example, obtained by transformation of the original query, would indeed normally imply that some time was spent by the professors in the country which issued their degree.

These inference techniques based on the principle of searching semantic analogies have been tested for a long time in the framework of the RESEDA project (<ZARR81>, <ZARR83a>, <ZARR84a>,

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

<ZARR84b>, <ZARR85>). RESEDA is an "intelligent" Information Retrieval system using techniques borrowed from Knowledge Engineering (Expert Systems and Knowledge Base Management Systems) in the field of complex biographical data management. This system is able to answer indirectly questions not having a direct answer, by using a first stage of inference procedures corresponding to the transformations. Moreover, the system is able to establish new causal links between statements represented in its own, "semantic" database, on the ground of "hypotheses", of a somewhat general nature, about the class of possible relationships. In this case, the result of the inference operations can thus modify, at least in principle, the very content of the database.

2. An informal overview of transformations

In this section, I will now give some precisions about the intuitive notion of transformation as introduced above.

Transformations are made up of a "left hand side" - formulation in a particular knowledge representation language of the linguistic expression which is to be transformed - and one or more "right hand sides" - representation in the same "metalanguage" of one or more linguistic expressions that will have to be substituted for the given one. These rules can be synthesized starting from their description in natural language, see <ZARR83b>, <FARI84>. A transformation can be formally expressed as :

A (left hand side) ⟶ B (right hand side)

The "transformation arrow" has a double meaning:

- operationally speaking, the arrow indicates the direction of the transformation : the left hand side A is dropped and replaced by the right hand side B ;
- The *normal* logical meaning of the arrow is that the information obtained through B implies the one obtained from A ; I will come back later, in this section, to this issue.

To keep to a first, extremely simple example, consider the transformation in Fig. 1, which is the formal translation in the RESEDA's metalanguage, <ZARR84a>, of the natural language "common sense" rule : If someone goes from one place to another, then he certainly has left his starting point.

t1) end+BE-PRESENT SUBJ *x* : *k* ⟶ MOVE SUBJ *x* : *k*
OBJ *x* : *l*

x = <human_being>
k, *l* = <location>
k ≠ *l*

Figure 1

In terms of the RESEDA metalanguage, BE-PRESENT and MOVE are "predicates" following the particular "case grammar" conception as developed by AI researchers from the ideas of Fillmore (<FILL68>, <FILL77>), see for example <BRUC75>, <ROSN80>, <CHAR81>. One or more "modulators", "end" in Fig. 1, may be attached to each predicate in order to inflect its basic semantic significance (the base meaning of "BE-PRESENT" is "to live in a place"). Each predicate is accompanied by "case slots" (SUBJect, OBJect, SOURCE, DESTination, MODALity, EVENT, etc.) which introduce its arguments, in our case the variable "*x*". In Fig. 1, this variable is followed by two location variables "*k*" - the starting point - and "*l*" - the arrival point. The values which can replace all these variables must obviously respect the constraints associated with the variables, see Fig. 1 ; each constraint is expressed as the "root" of a sub-tree ("isa" and "prototype-instance" relationships, see also Fig. 7 in the next section) that describes the various concrete possibilities of substitution.

In the case of Fig. 1, we have an implication B ⟹ A (⟹ is the "implication arrow"), that can be considered as *always* true. In fact, any information about some human being *x* having moved from *k* to *l* is always, at the same time, a response to any query about the possibility of him no longer being present at location *k*, at least if one considers the existence of a temporal continuity between events matching parts A and B of the transformation. Another implication of this type may be found by examining the transformation t2 in Fig. 2, one of the two transformations necessary to process the second example of the "Introduction". The association of the two modulators "abs = absolutely" and "begin" with the predicate BE-PRESENT is the RESEDA translation of the idea "to be born" (beginning, without reference to a previous history, of the existence at the location "*k*"). Being born in a place always implies that one has been present at that location at least for a short time.

t2) BE-PRESENT SUBJ *x* : *k* ⟶ begin+abs+BE-PRESENT
SUBJ *x* : *k*

x = <human_being>
k = <location>

Figure 2

Still in the domain of "true implication", at least in the context of a well determined "world", we face sometimes situations where it is not possible to state clearly that the right hand side B should imply the left hand side A rather then the contrary ; in this case, that we call "equivalence", we will have arrows in both directions (operatively speaking A ⟺ B authorizes us to replace A by B and the inverse is equally true). An example of an equivalence is given in Fig. 3.

t3) BE-AFFECTED-BY ⟺ BE-AFFECTED-BY
SUBJ *p* SUBJ *x*
OBJ *x* (SPECIF *q*) OBJ *r*
SOURCE *p*

x = <human_being>
p = <social_body>
q = <permanent_post>
r = <permanent_salary>

Figure 3

The common sense rule underlying this formalism is : To know if someone (*x*) is employed, in a permanent way, by an organization (*p*) - the organization *p* is

```
t4)   BE-PRESENT  SUBJ  x : z        ⟶   BE-AFFECTED-BY  SUBJ  x
                                                          OBJ   p
                                                       SOURCE   q : z

      x = <human_being>
      p = <official_document>
      q = <authority>
      z = <location>

t5)   ment+BE-AFFECTED-BY  SUBJ  x                ⟶   BE-AFFECTED-BY  SUBJ  r  (SPECIF p)
                           OBJ   p                                      OBJ   x  (SPECIF s)
                         MODAL   proficiency  (SPECIF q)
      x = <human_being>
      p = <technical_domain>
      q = <degree_of>
      r = <subpart_of_a_social_body>
      s = <post>

t6)   ment+BE-AFFECTED-BY  SUBJ  x                ⟶   PRODUCE  SUBJ  x
                           OBJ   p                             OBJ   r
                         MODAL   proficiency  (SPECIF q)       EVENT  s  (SPECIF p)
      x = <human_being>
      p = <technical_domain>
      q = <degree_of>
      r = <professional_activity>
      s = <industrial_project>

t7)   end+BE-AFFECTED-BY  SUBJ  p                  ⟶   PRODUCE  SUBJ  x
                          OBJ   x  (SPECIF q)                   OBJ   r
                                                                ARG   s

      x = <human_being>
      p = <social_body>
      q = <post>
      r = <professional_fault>
      s = <activity_having_p_as_source>
```

Figure 4

"augmented", BE-AFFECTED-BY, by the person x - one might check, amongst other things, whether x regularly receives money (r) from this organization (or vice versa). Notice, in transformation t3, the different syntactic construction associated to the predicate BE-AFFECTED-BY depending on whether x is part of the organism p (left hand side) or whether he is in possession of an object r (right hand side).

The case of always true implications, however, is not the most frequent. Most transformations found in real world applications are in fact "modalized implications" (noted " B *⟶ A ", which means "it is possible that B implies A"), see for example <KODR85>. Examples of this type of transformations may be found in Fig. 4.

The transformation t4 says : If someone receives an official document from an official authority, then it is possible that he has been physically present at that moment in the place where the authority is located (c.f. the second example in the "Introduction"). This rule, for example, is *not always valid* in the case of a university diploma (it could be obtained in a correspondence school, etc.). Nevertheless, it is possible to see that, in this case, the "semantic distance" between a "sure" implication and a "modalized" one is not considerable as it is always possible to change t4 into a "true" transformation by the addition of a few constraints on the variable p, for instance $p \neq$ <obtainable_by_correspondence_document>, etc.

Transformations t5 and t6 are necessary to process the first example in the "Introduction". Transformation t5 links "having some knowledge in AI (p)" to "belong to the AI department (r) of the company" (ment+BE-AFFECTED-BY = to know, to be expert in). Transformation t6 links "having some knowledge in AI" to "having a professional activity (EVENT = in the context of) in a project (s) related to AI (SPECIF p)". In both cases, the staff not concerned directly with AI (ex: secretaries) will satisfy the right hand side of the transformation and the constraints associated with the variables. To make t5 and t6 "sure", by adding more constraints, would be much more delicate than in the case of t4.

Transformation t7 is the best example of a truly modalized rule : there is absolutely no guarantee that committing a professional fault does necessarily imply getting fired (end+BE-AFFECTED-BY).

I will conclude this section by saying that t1, t2, t4, t5, t6, t7 can, in principle, be reversed, i.e. be used in the form B ⟶ A. The corresponding implications, which are always modalized, therefore take the form " A *⟶ B " with different plausibilities, very low for the inverses of t2 and t4 (to be physically present in a place, generally has no link with being born there or being granted a diploma), and very high for the inverse of t7 (it is always possible to explain one's getting fired

by one's making a professional fault).

3. The architecture of the system

Fig. 5 contains a simplified flowchart of the system, under construction, capable of the rational behaviour explained in the preceding sections.

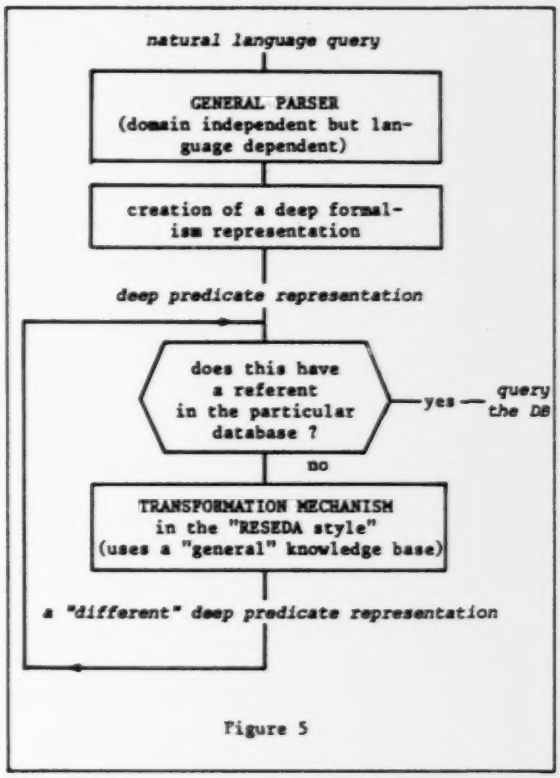


Figure 5

First of all the natural language (i.e. French) query is parsed and partially disambiguated, using structure rules, <GROS75>, and a syntactical grammar of French. The parser will not use any particular semantic knowledge which is typical of the chosen application domain. In some cases, the parser resorts to simple clarifying dialogues. As the result of this analysis, a predicate representation is produced. It carries the meaning of the user's query.

The system has at its disposal a model of the structure of the database being accessed (a special predicate calculus description, close to the conceptual schema), so it can check whether this query is a meaningful request to the database. If it is meaningful, then the system translates the deep predicate representation into a corresponding formal query language, giving rise for example to an SQL formula.

However, in many cases, the query will not map directly onto the database.

Instead of simply telling the user that his question is not "the question to ask", we will try to transform his request into something meaningful in the particular database and *semantically close* to the original request (c.f. the examples mentioned in the sections above).

The transformations necessary for this type of operation form the core of the system's knowledge base.

From a technical point of view, the overall rule base is divided into a number of smaller knowledge bases, as illustrated in Fig. 6.

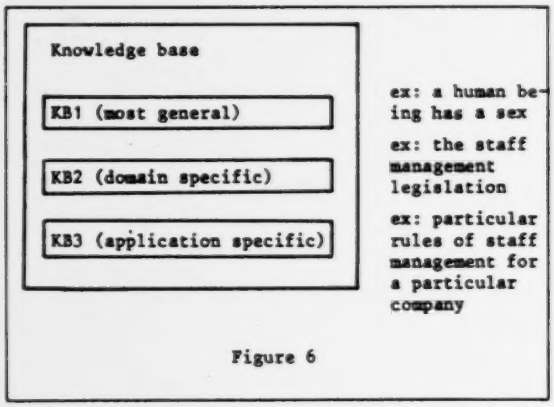

Knowledge base
KB1 (most general)
KB2 (domain specific)
KB3 (application specific)
ex: a human being has a sex
ex: the staff management legislation
ex: particular rules of staff management for a particular company

Figure 6

KB1 contains bits of knowledge that are usable in any application (of course, it will never be complete).

KB2 contains general knowledge for a particular domain. We hope to develop a certain number of KB2s (with a common KB1 nucleus) for each real world domain that our system will have to deal with.

Each KB1 + KB2 couple is transportable "as is", as long as we do not switch to a complete different domain.

Note that *KB3* is not specific to a particular database ; the system can use the same KB3 in order to access different personnel databases of the same company.

The *model of the database* itself (the most specific level of knowledge) is not included here since it does not take part directly in the transformation process.

Any one of the KBs mentioned above may contain two types of knowledge : factual knowledge and rules.

Factual knowledge. A fact such that "a university is in a city ...", which is a piece of factual knowledge, is represented by the highlighted part of Fig. 7. This network describes "specific-generic" ("isa" links) and "prototype-instance" types of relations, used as matching criteria when applying rules. Other specific relations used in the rules, for example topological ones, are also present in the network.

Rules. Two types of rules can be distinguished :

- transformations rules ;
- standardization rules.

For a description of the (external) formalism used to formulate transformation rules, see section 2.

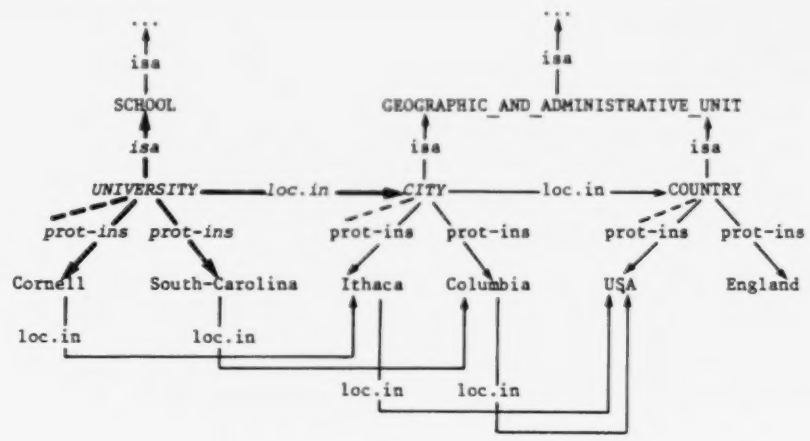


Figure 7

Standardization rules are used to translate the parser's output (containing surface verbs) into a more canonical form, i.e. usable by transformation rules. All of the standardization rules belong to the KB1 level, as they are absolutely general.

The *inference engine* has to create all possible new formal expressions consistent with the transformations considered and with the particular formulation of their left hand side.

As far as the general architecture of the engine is concerned, it should not be very different from the one used in RESEDA - see <ZARR84b>, <ZARR85> for the description of the later one. Its functioning will therefore consist of two main phases :

- A bottom-up phase which involves going from the *particular* expression to transform (where a certain number of variables have already been instantiated) and selecting in the knowledge base one or more "rule heads" (left hand side of a transformation) which define a *generic* class encompassing the expression in question : this is the *selection phase*. The correspondence between the expression to transform and the rules heads is determined by examining the elements, modulators, predicates and case slots, which provide the basic structure of a coded formulation in the RESEDA's metalanguage, thus discarding the data formed by the terms of the lexicon, or the variables carrying restrictions which can replace them.
- A top-down phase (corresponding to the top-down functioning of PROLOG interpreters), during which the program corresponding to the selected rule or rules is executed by generating *particular* right hand sides (at least a part of the values deriving from the expression to be transformed are transmitted to the right hand side) and thus new specific queries to the database : this is the *execution phase*.

Fig. 8, <EUZE84b>, briefly summarizes the algorithm of the engine, assuming the adoption of backtracking techniques and an exhaustive search strategy.

"Backtrack activation points" can be :

- located inside the left hand side of an already examined rule, in case of variables which can assume new values ;
- located outside the rules, and in this case related to the rules not yet utilized belonging to the selected set.

We may observe that the succesful mapping of a query does not necessarily mean the interruption of the transformation process. In the case of an exhaustive search, this process can continue in order to obtain all the possible answers coherent with the concept of semantic proximity. This method enables us to have a full panorama of the successes and failures associated with the different branches of the "choice tree" constructed by the inference engine ; obviously, this can be computationally expensive.

The transformation process can be halted in different ways :

- no transformation applies ;
- there are not any backtrack points left ;
- halt ordered by the user ;
- etc.

4. Conclusion

This paper describes the development of "intelligent" tools aimed at improving the retrieval facilities from large relational databases. The inference mechanism adopted allows, when needed, to automatically convert an input query into another one which is "semantically close" *without accessing the database*. According to RESEDA's theory, "semantically close" means that the answer to the transformed query implies what could have been the answer to the original question.

This project is conducted at ERLI ("Etudes et

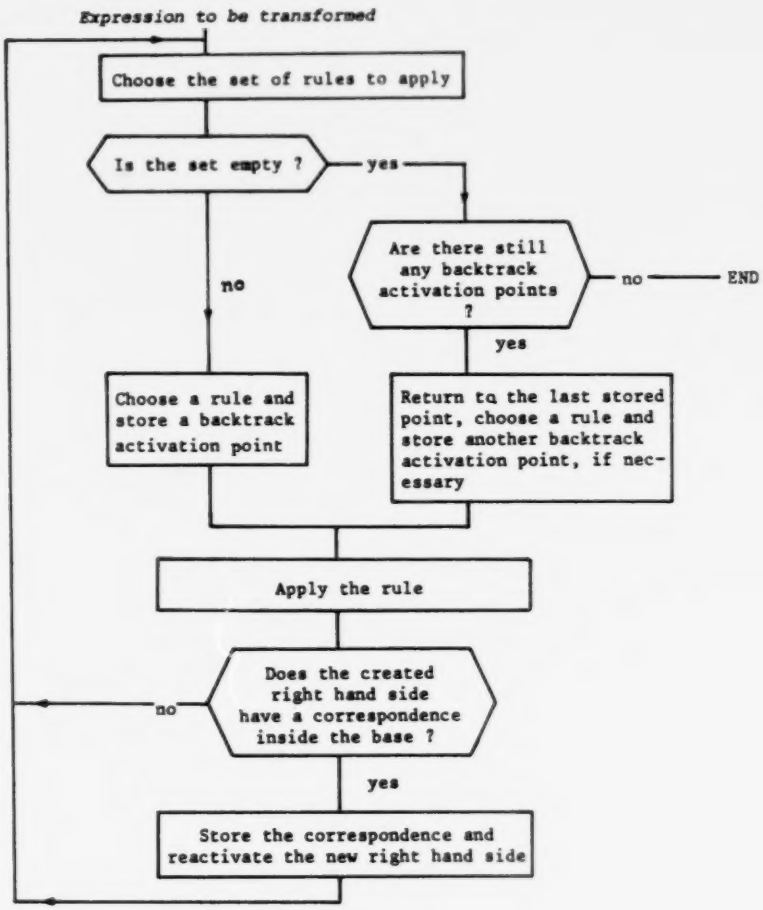

Expression to be transformed
Choose the set of rules to apply
Is the set empty ?
yes
no
Are there still any backtrack activation points ?
no
END
yes
Choose a rule and store a backtrack activation point
Return to the last stored point, choose a rule and store another backtrack activation point, if necessary
Apply the rule
Does the created right hand side have a correspondence inside the base ?
no
yes
Store the correspondence and reactivate the new right hand side

Figure 8

Recherche en Linguistique et Informatique"), a private commercial company involved in natural language processing (natural language interfaces, terminology and dictionary management software) since 1977, see <NORM84a>, <NORM84b>. A preliminary phase of this research was supported in part by the French "Agence de l'Informatique", research contract ADI-84-068.

ACKNOWLEDGEMENTS

The author is indebted to Mr. Antoine Ogonowski of ERLI for many useful discussions during the preparation of this paper.

REFERENCES

<BRUC75> Bruce, B. (1975) "Case Systems for Natural Language", *Artificial Intelligence*, VI, 327-360.

<CHAR81> Charniak, E. (1981) "The Case-Slot Identity Theory", *Cognitive Science*, V, 285-292.

<EUZE84a> Euzenat, B., Normier, B., Ogonowski, A., and Zarri, G.P. (1984) *Les mécanismes d'inference dans l'interrogation de bases de données relationnelles en langage naturel* (Convention ADI-84-068, rapport intermédiaire). Charenton: ERLI.

<EUZE84b> Euzenat, B., Normier, B., Ogonowski, A., and Zarri, G.P. (1984) *Les mécanismes d'inférence dans l'interrogation de bases de données relationnelles en langage naturel* (Convention ADI-84-068, rapport final). Charenton: ERLI.

<EUZE85> Euzenat, B., Normier, B., Ogonowski, A., and Zarri, G.P. (1985) "SAPHIR+RESEDA, a New Approach to Intelligent Data Base Access", in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence - IJCAI/85*. Los Altos: Morgan Kaufmann.

<FARI84> Faribault, M., Léon, Jacqueline, Meissonnier, V., Memmi, D., and Zarri, G.P. (1984) "From Natural Language to a Canonical Representation of the Corresponding Semantic Relationships", in *Cy-*

bernetics and Systems Research 2, Trappl, R., ed. Amsterdam: North-Holland.

<FILL68> Fillmore, C.J. (1968) "The Case for Case", in *Universals in Linguistic Theory*, Bach, E., and Harms, R.T., eds. New York: Holt, Rinehart and Winston.

<FILL77> Fillmore, C.J. (1977) "The Case for Case Reopened", in *Syntax and Semantics 8 : Grammatical Relations*, Cole P., and Saddock, J.M., eds. New York: Academic Press.

<FINI79> Finin, T., Goodman, G., and Tennant, H. (1979) *JETS : Achieving Completeness Through Coverage and Closure* (Working Paper n° 19). Urbana: University of Illinois Coordinated Science Laboratory.

<GROS75> Gross, M. (1975) *Méthodes en syntaxe*. Paris: Hermann.

<KAPL82> Kaplan, S.J. (1982) "Cooperative Responses from a Portable Natural Language Query System", *Artificial Intelligence*, IXX, 165-187.

<KELL78> Kellog, C., Klahr, P., and Travis, L. (1978) "Deductive Planning and Pathfinding for Relational Databases", in *Logic and Data Bases*, Gallaire, H., and Minker, J., eds. New York: Plenum Press.

<KODR85> Kodratoff, Y., Perdrix, H., and Franova, Marta (1985) "Traitement symbolique du raisonnement incertain", in *Proceedings of the Congress on Hardware, Software Components and Architectures for the 5th Generation*. Paris: AFCET.

<NORM84a> Normier, B. (1984) *SAPHIR - Présentation générale*. Charenton: ERLI.

<NORM84b> Normier, B., Aubert, M.E., Clemencin, G., Euzenat, B., Lacep, S., Lellouche, N., Ogonowski, A., and Vega, J. (1984) *Extension de la gestion du dialogue* (Marché DRET 83/234, rapport final). Charenton: ERLI.

<PYLY85> Pylyshyn, Z.W., and Kittredge, R.I. (1985) "Databases and Natural Language Processing", *Quarterly Bulletin of the IEEE Computer Society Technical Committee on Database Engineering*, VIII, n° 3, 2-9.

<ROSN80> Rosner, M., and Somers, H.L. (1980) *Case in Linguistics and Cognitive Science* (Working Paper n° 40). Gèneve: Institut pour les Etudes Sémantiques et Cognitives.

<WAHL84> Wahlster, W. (1984) "Cooperative Access Systems", *Future Generations Computer Systems*, I, 103-111.

<WEBB84> Webber, Bonnie Lynn, and Finin, T. (1984) "In Response : Next Steps in Natural Language Interaction", in *Artificial Intelligence Applications for Business*, Reitman, W., ed. Norwood: Ablex.

<WOOD84> Woods, W.A. (1984) "Natural Language Communication With Machines : An Ongoing Goal", in *Artificial Intelligence Applications for Business*, Reitman, W., ed. Norwood: Ablex.

<ZARR81> Zarri, G.P. (1981) "Building the Inference Component of an Historical Information Retrieval System", in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence - IJCAI/81*. Los Altos: Morgan Kaufmann.

<ZARR83a> Zarri, G.P. (1983) "An Outline of the Representation and Use of Temporal Data in the RESEDA System", *Information Technology : Research and Development*, II, 89-108.

<ZARR83b> Zarri, G.P. (1983) "Automatic Representation of the Semantic Relationships Corresponding to a French Surface Expression", in *Proceedings of the Conference on Applied Natural Language Processing*. Menlo Park: Association for Computational Linguistics.

<ZARR84a> Zarri, G.P. (1984) "Expert Systems and Information Retrieval : An Experiment in the Domain of Biographical Data Management", in *Developments in Expert Systems*, Coombs, M.J., ed. London: Academic Press.

<ZARR84b> Zarri, G.P. (1984) "Some Remarks About the Inference Techniques of RESEDA, an 'Intelligent' Information Retrieval System", in *Research and Development in Information Retrieval - Proceedings of the Third Joint BCS and ACM Symposium*, van Rijsbergen, C.J., ed. Cambridge: University Press.

<ZARR85> Zarri, G.P. (1985) "Inference Techniques for Intelligent Information Retrieval", in *Computing Power and Legal Reasoning*, Walter, C., ed. St Paul: West Publishing Company.

A MACHINE LEARNING APPROACH TO INFORMATION RETRIEVAL

S.K.M. WONG and W. ZIARKO

Department of Computer Science, University of Regina,
Regina, Saskatchewan, Canada, S4S 0A2

1. INTRODUCTION

The fundamental problem in Information Retrieval (IR) is to identify the relevant documents from the nonrelevant ones in a collection of documents according to a particular user's information needs. One of the major difficulties in modelling information retrieval is to choose an appropriate (knowledge) representation of the content of an individual document. For example, it is common to describe each document by a set of (weighted) index terms or keywords obtained from an automatic indexing scheme [1, 2, 3]. Since these index terms (or some other similar "constructs") provide us only with partial knowledge about the contents of the documents, it is unrealistic to expect that the system would identify without uncertainty only those documents the user needs. Thus, any relevance judgment based on the surrogate documents and some highly model dependent retrieval strategy is bound to be uncertain. In this regard, the search strategy adopted in the standard Boolean model, for example, used in most commercial systems is generally considered to be too restrictive. On the other hand, a *soft* matching function is provided by the vector-based [2, 3, 4] and the fuzzy set [5, 6] models to rank documents with respect to the degree of similarity between a surrogate document and the query. In these approaches, it is believed that the query formulated by the user (using whatever query language provided by a particular IR model) can in fact accurately reflect a user's information requirements. However, in practice, more often than not a user may not be able to describe in a precise way the characteristics of the relevant information items even with a query language of sufficient expressive power [7]. This drawback, to some extent, can be remedied by incorporating some intuitive relevance feedback procedure [8] to improve the query formulation.

So far the majority of research work in information retrieval is largely non-probabilistic even though significant headway has been made with probabilistic methods [9]. One of the main reasons why the probabilistic model has not been widely accepted is, perhaps, due to its computational complexity. Although the conditional probabilities can be easily computed when the index terms are assumed to be independent (or only a few of them are dependent), the complexity of the computation would grow exponentially with the increase of the number of dependent terms [9]. To cope with this problem, several approximate methods were proposed by different authors [10-15]. In essence, the computation of the probability distributions in all these methods is based on the probabilities of occurrence of the individual terms. Regardless of which method is used to calculate the probabilities, the objective of the probabilistic method is to derive (based on the user relevance judgment) a set of decision rules in terms of the binary or weighted index terms. These rules are subsequently used to predict, with some certainty, the relevance status of an individual document.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

The probabilistic approach to information retrieval can, therefore, be viewed as a special solution to the problem of "concept" learning (or inductive learning) in artificial intelligence. The concept we are trying to learn in information retrieval is the user concept of relevant documents. The set of decision rules derived from the probabilistic model provides the description of such a concept. The main objective of this paper is to apply the learning model [16] we proposed to generate a generalized set of decision rules to characterize the relevant documents. The question that naturally arises is how are these two approaches related to each other. To answer this question, we refer to the fundamental objective of inductive learning, which is the ability to "make accurate generalizations from a few scattered facts or to discover patterns in seemingly chaotic collections of observations" [17]. This ability is absent in the probabilistic model since no attempt is made to generalize the decision rules from the document descriptions. Consequently, in some situations, making predictions based on these restricted rules may not be possible. Another problem is that the number of decision rules could be the same as the number of distinct documents, making it impracticable to maintain such a large number of rules for future reference. Thus, whatever has been learned about the user concept of relevant documents would not be retained in the system. However, this information of user profiles may prove to be extremely valuable in future interactions between the user and the system. For example, the user may query the system by identifying one of these stored profiles, thus freeing the system from recomputing the decision rules.

In contrast to the probabilistic approach, an effective inductive learning algorithm should be able to select a subset of dominant index terms to approximately characterize the user concept by a minimal set of generalized decision rules. This would make it possible to store the user profiles as a part of the system. In this paper, we would like to present an information retrieval model based on the principles of learning by *examples*. "By examples" refer to those documents which have been reviewed and evaluated by the user as being relevant or nonrelevant. The formal tool which underlies our model is the notion of probabilistic approximate classification [16], which we introduced recently as an extension of the rough set theory proposed by Pawlak [18, 19]. Within this framework, we have developed a Probabilistic Learning Algorithm (*PLA*) [16] which can be directly applied to design a truly adaptive information retrieval system. (This algorithm has been successfully implemented with LISP in a microprocessor.)

In the following sections, we first give a brief introduction to inductive learning in the context of information retrieval before we discuss the basic notions of the probabilistic approximate classification and the knowledge representation system. Based on these notions, we then show how decision rules to support the retrieval of relevant documents can be constructed from our learning model. Finally, we present an algorithmic description of the interactive learning process which can lead to fast retrieval of the majority of relevant documents.

Throughout this paper, we attempt to illustrate our main ideas with some simple examples whenever possible, which will perhaps make the discussions less formal but hopefully more readable.

2. INDUCTIVE LEARNING

By inductive learning from the information retrieval perspective, we mean learning and refining decision rules to separate relevant from nonrelevant documents in a collection, based on the examples of decisions made by the system user.

The subject of machine learning has received considerable attention in recent years. Inductive learning is perhaps the oldest and best understood problem in artificial intelligence. Some of the more recent research efforts made in this area can be found in [20]. For example, Quinlan [21] suggested an inductive algorithm based on the statistical theory of information originally proposed by Shannon [22]. The entropy function is used as a measure of uncertainty in the classification of objects characterized by *features* (attributes and attribute values). On the other hand, Pawlak [19] indicated that the principles of inductive learning may be formulated in a unified way within the framework of the theory of rough sets.

The central question in the design of an adaptive information retrieval system is whether we can identify the user concept of relevant documents solely on the basis of the syntactic knowledge (e.g. terms and their weights). We address this question by introducing the notion of probabilistic approximate classification to determine the extent to which we can identify a user's concept in a knowledge representation system.

3. KNOWLEDGE REPRESENTATION SYSTEM IN INFORMATION RETRIEVAL

The basic component of a knowledge representation system [19] is a set of documents. We assume that our knowledge about these documents is expressed through assignment of some index terms and their weights to the individual documents.

Let $S = \langle D, T, REL, V_T, V_{REL}, f \rangle$ denote a knowledge representation system, where D is a set of documents, T is the set of index terms used to index the documents in D, and REL is an attribute which describes the user judgment about the relevance status of a document. Both the index terms and REL are referred to as document attributes. The knowledge function f in S can be defined as follows:

$$f: D \times T \rightarrow V_T = \bigcup_{t \in T} V_t ,$$
$$f: D \times \{REL\} \rightarrow V_{REL} ,$$

such that for every $d \in D$, $f(d, a) \in V_a$, where V_a is the domain of the attribute $a \in T \cup \{REL\}$. For example, we may assume that $V_a = \{0,1\}$. In this case, each index term is either present or absent in a document, and each document is either relevant or nonrelevant from a user's point of view. However, it should be emphasized that it is not necessary to assume binary term weights in our model.

In information retrieval the user, after viewing any subset of documents, is able to classify these documents according to their relevance status. The system, on the other hand, can classify the documents based only on the values (weights) of the index terms. The objective in inductive learning is to "teach" the system by using a properly chosen set of *training examples* such that it can approximately reproduce the user classification.

Example 1.

Consider the following set of documents (which may be regarded as a set of training examples) given in Table 1.

In Table 1, each document in $D = \{d_1, d_2, \ldots, d_8\}$ is described by the set of index terms, $T = \{MATHEMATICS, PHYSICS, ALGEBRA\}$ with attribute values $V_{MAT} = V_{PHY} = V_{ALG} = \{0,1\}$. According to the user judgement, each document belongs to either class "1" (relevant) or class "0" (nonrelevant), i.e. $V_{REL} = \{0,1\}$. The values of the information function $f(d, a)$ are summarized in Table 1 (e.g. $f(d_2, MAT) = 0$ and $f(d_4, ALG) = 1$, etc.). □

D	MAT	PHY	ALG	REL
d_1	1	1	1	0
d_2	0	1	1	0
d_3	0	1	0	0
d_4	0	1	1	1
d_5	1	0	1	1
d_6	1	0	1	0
d_7	0	0	0	0
d_8	0	0	1	1
d_9	1	0	0	0

Table 1. An example of a knowledge representation system.

4. APPROXIMATE CHARACTERIZATION OF THE SET OF RELEVANT DOCUMENTS

In the knowledge representation system we assume that each document can be characterized by some selected index terms. In general, information about document content expressed in this way is not sufficient to characterize the documents uniquely, as any two documents are indistinguishable from one another whenever they assume the same values for all the index terms under consideration. This means that we may not be able to distinguish all the documents solely by means of the admitted terms. For example, we can only approximately define the membership of the subset of documents $E = \{d_6, d_8, d_9\}$ in Table 1 by the values of the term MAT. To state this problem in a more precise way, we need to introduce the notions of partitions and the probabilistic approximate classification.

4.1. Partitions Induced by Index Terms

In a knowledge representation system $S = \langle D, T, REL, V_T, V_{REL}, f \rangle$, for any subset B of index terms in T, we can define an equivalence relation $\Theta(B)$ on $D = \{d_1, d_2, \ldots, d_m\}$ such that:

$$(d_i, d_j) \in \Theta(B) \text{ iff } f(d_i, b) = f(d_j, b), \text{ for } \textit{every } b \in B.$$

Let Θ_Π be an equivalence relation induced by a partition Π on D into a family of disjoint elementary sets, $\Pi = \{X_1, X_2, \ldots, X_n\}$. In subsequent discussions, we will consider only those partitions such that every elementary set $X \in \Pi$ is identical to an equivalence class of the equivalence relation $\Theta(B_X)$ defined by some subset of terms $B_X = \{b_1, b_2, \ldots, b_k\} \subseteq T$. Thus, each class $X \in \Pi$ can be uniquely described by a conjunction of the attribute values of some B_X, namely,

$$Des(X) = Des(d) = f(d, b_1) \wedge f(d, b_2) \wedge \cdots \wedge f(d, b_k),$$

for all $d \in X$. $Des(X)$ is referred to as the description of the elementary set X of the partition Π. Among all these partitions induced by the index terms, the atomic partition, for example, defined by $B_X = T$ for every elementary set X, has the finest *granularity* in a given knowledge representation system.

Example 2.

The elementary sets of a partition $\Pi = \{X_1, X_2, X_3\}$ of the documents in Table 1 can be defined by the values of the index terms as follows:

$$X_1 = \{d_1, d_2, d_4, d_5, d_6, d_8\},$$
$$X_2 = \{d_3, d_7\},$$
$$X_3 = \{d_9\},$$

$$Des(X_1) = (ALG := 1),$$
$$Des(X_2) = (MAT := 0) \wedge (ALG := 0),$$
$$Des(X_3) = (MAT := 1) \wedge (PHY := 0) \wedge (ALG := 0).$$

Note that $B_{X_1} = \{ALG\}$, $B_{X_2} = \{MAT, ALG\}$, and $B_{X_3} = \{MAT, PHY, ALG\}$. □

UBS 241

4.2. Probabilistic Approximate Classification

Let $E \subseteq D$ be a subset of documents in a knowledge representation system S, identified by a user as being relevant. Given a partition $\Pi = \{X_1, X_2, \ldots, X_n\}$ on D induced by some index terms, our objective is to classify the set E, based on the descriptions of the elementary sets of Π. For this purpose, we first introduce the notion of a probabilistic approximation space $A = \langle D, \Pi, P \rangle$, where P is a probability measure defined on the subsets of documents of D.

In order to measure how well the set of descriptions, $\{Des(X_i) \mid X_i \in \Pi\}$, can actually specify the membership of the set of relevant documents E, we introduce the important notion of probabilistic approximate classification as follows:

(i) The A-lower approximation of the set of relevant documents E in $A = \langle D, \Pi, P \rangle$ is defined as

$$\underline{A}(E) = \{d \in D \mid d \in X_i \text{ and } P(E \mid X_i) > \frac{1}{2}\}.$$

$\underline{A}(E)$ is in fact the union of all those elementary sets $X_i \in \Pi$ with conditional probabilities $P(E \mid X_i) > \frac{1}{2}$. We also refer to $\underline{A}(E)$ as the positive region $POS(E)$ of the set of relevant documents.

(ii) The A-upper approximation of E is defined as

$$\overline{A}(E) = \{d \in D \mid d \in X_i \text{ and } P(E \mid X_i) \geq \frac{1}{2}\}.$$

$\overline{A}(E)$ is the union of all those elementary sets $X_i \in \Pi$ with conditional probabilities $P(E \mid X_i) \geq \frac{1}{2}$.

(iii) The set $BND(E) = \overline{A}(E) - \underline{A}(E)$ is called the doubtful region or the boundary region of E. That is, $BND(E)$ consists of all those elements in D with conditional probabilities $P(E \mid X_i) = \frac{1}{2}$.

(iv) The set $NEG(E) = D - \overline{A}(E)$ is called the negative region of E.

It should be emphasized here that the uncertainty is maximal, when trying to determine whether or not any document in the boundary region $BND(E)$ is relevant, based only on the descriptions of the elementary sets of Π. On the other hand, we can determine with lesser uncertainty if a document in the positive region $POS(E)$ is relevant (or if a document in the negative region $NEG(E)$ is nonrelevant).

Since a set $E \subseteq D$ may or may not have clearly defined boundaries in an approximation space based on the descriptions of the elementary sets of a partition, this observation, in fact, leads to the notion of *rough sets* introduced by Pawlak [17].

Example 3.

Let Π be a partition of the set of documents given in Table 1, induced by the subset of index terms $B = \{MAT, ALG\}$. This partition $\Pi = \{X_1, X_2, X_3, X_4\}$,

$$\begin{aligned} X_1 &= \{d_1, d_5, d_6\}, & DES(X_1) &= (MAT := 1) \wedge (ALG := 1), \\ X_2 &= \{d_2, d_8, d_9\}, & DES(X_2) &= (MAT := 0) \wedge (ALG := 1), \\ X_3 &= \{d_3, d_7\}, & DES(X_3) &= (MAT := 0) \wedge (ALG := 0), \\ X_4 &= \{d_4\}, & DES(X_4) &= (MAT := 1) \wedge (ALG := 0), \end{aligned}$$

defines a probabilistic approximation space $\langle D, \Pi, P \rangle$. In order to find the lower and upper approximations of the set of relevant documents $E = \{d_4, d_5, d_6\}$, we need to compute the conditional probability $P(E \mid X_i)$ for each of the elementary sets in Π. It can be easily verified from Table 1 that

$$P(E \mid X_1) = \frac{|E \cap X_1|}{|X_1|} = \frac{1}{3},$$

$$P(E \mid X_2) = \frac{|E \cap X_2|}{|X_2|} = \frac{2}{3},$$

$$P(E \mid X_3) = \frac{|E \cap X_3|}{|X_3|} = 0,$$

$$P(E \mid X_4) = \frac{|E \cap X_4|}{|X_4|} = 0,$$

where $|\ldots|$ denotes the cardinality of a set. Therefore, the lower and upper approximations of E are given respectively by

$$\underline{A}(E) = X_2 \quad \text{and} \quad \overline{A}(E) = X_2,$$

which implies that

$$POS(E) = \underline{A}(E) = X_2,$$

$$BND(E) = \overline{A}(E) - \underline{A}(E) = X_2 - X_2 = \phi,$$

$$NEG(E) = D - \overline{A}(E) = X_1 \cup X_3 \cup X_4.$$

This means that we can predict that document d is likely to be relevant only if it matches the description $DES(d) = DES(X_2)$. □

4.3. Decision Rules with Certainty Measures

The relationship between the set of relevant documents and the elementary sets of a partition in a probabilistic approximation space can be formally expressed in the form of decision rules. By the nature of the probabilistic approximate classification, there is some degree of uncertainty associated with each of these rules. The decision rules in our model are conditional statements of the kind " if <condition> then <conclusion> " assigned to each of the elementary sets of the approximation space. The <condition> part of each rule is the description of the corresponding elementary set and the <conclusion> is the prediction stating whether a document satisfying the given description is relevant or nonrelevant. We also have a special kind of <conclusion> = *unknown* whenever the system does not have enough information to draw any definite conclusion. In such a case, a further refinement of the approximation space may be necessary.

Given the set of relevant documents E in an approximation space $\langle D, \Pi, P \rangle$, we can define a decision rule r_i for each elementary set $X_i \in \Pi = \{X_1, X_2, \ldots, X_n\}$ in the following way:

$$r_i : DES(X_i) \rightarrow \begin{cases} relevant & \text{if } X_i \in POS(E), \\ nonrelevant & \text{if } X_i \in NEG(E), \\ unknown & \text{if } X_i \in BND(E). \end{cases}$$

This means that if a document d satisfies the description $DES(X_i)$ (i.e. $DES(d) = DES(X_i)$) and $d \in POS(E)$, then d is possibly relevant with some statistical certainty. Similarly, if $d \in NEG(E)$, then d is possibly nonrelevant. It is clear that an arbitrary decision must be made whenever $d \in BND(E)$ and the uncertainty of such a decision is maximal.

Therefore, the positive and negative regions of E characterize the *certainty area* of the universe of discourse. If a document belongs to the positive region, a decision can be made with some degree of certainty that it is relevant; likewise, if a document belongs to the negative region, we can say with some certainty that it is nonrelevant. Nothing can be said if a document belongs to the doubtful region of the relevant documents.

We showed in [16] that the entropy function from the information theory [22] provides a statistical measure of certainty associated with the predictions based on our decision rules. However, in the particular case of having only two concepts (*relevant* and *nonrelevant*), it can be proven that the parameter,

$$c_i = \max(P(E \mid X_i),\ 1 - P(E \mid X_i)),$$

provides an equivalent certainty measure of the decision rule r_i.

Example 4.

From the positive and negative regions of E obtained in Example 3, we arrive at the following decision rules with certainty measures as defined above:

$$r_1 : (MAT \coloneqq 1) \wedge (ALG \coloneqq 1) \xrightarrow{c_1 = 0.66} nonrelevant$$

$$r_2 : (MAT \coloneqq 0) \wedge (ALG \coloneqq 1) \xrightarrow{c_2 = 0.66} relevant$$

$$r_3 : (MAT \coloneqq 0) \wedge (ALG \coloneqq 0) \xrightarrow{c_3 = 1.0} nonrelevant$$

$$r_4 : (MAT \coloneqq 1) \wedge (ALG \coloneqq 0) \xrightarrow{c_4 = 1.0} nonrelevant \quad \square$$

5. GENERALIZATION OF DECISION RULES

The objective of inductive learning is to infer attribute dependencies from a small number of sample observations (decisions). In our case, we are specifically looking for dependencies between the index terms and the user judgment attribute *REL* in a knowledge representation system. The attribute dependencies inferred from the sample decisions can be expressed as a collection of decision rules, as described in the last section. The real problem, however, lies not in just finding attribute dependencies, but primarily in searching for dominant and nonredundant attributes to characterize the documents judged as being relevant by the user. To illustrate this problem, let us consider the following example.

Example 5.

Let Π_1 be an atomic partition (induced by all available index terms) of the set of documents in Table 1, namely,

$$\Pi_1 = \{X_{11}, X_{12}, X_{13}, X_{14}, X_{15}, X_{16}, X_{17}\},$$

where

$$X_{11} = \{d_1\}, \quad X_{12} = \{d_2, d_4\}, \quad X_{13} = \{d_3\}, \quad X_{14} = \{d_5, d_6\},$$
$$X_{15} = \{d_7\}, \quad X_{16} = \{d_8\}, \quad X_{17} = \{d_9\}.$$

The descriptions of the above elementary sets are given by:

$$DES(X_{11}) = (MAT \coloneqq 1) \wedge (PHY \coloneqq 1) \wedge (ALG \coloneqq 1)$$
$$DES(X_{12}) = (MAT \coloneqq 0) \wedge (PHY \coloneqq 1) \wedge (ALG \coloneqq 1)$$
$$DES(X_{13}) = (MAT \coloneqq 0) \wedge (PHY \coloneqq 1) \wedge (ALG \coloneqq 0)$$
$$DES(X_{14}) = (MAT \coloneqq 1) \wedge (PHY \coloneqq 0) \wedge (ALG \coloneqq 1)$$
$$DES(X_{15}) = (MAT \coloneqq 0) \wedge (PHY \coloneqq 0) \wedge (ALG \coloneqq 0)$$
$$DES(X_{16}) = (MAT \coloneqq 0) \wedge (PHY \coloneqq 0) \wedge (ALG \coloneqq 1)$$
$$DES(X_{17}) = (MAT \coloneqq 1) \wedge (PHY \coloneqq 0) \wedge (ALG \coloneqq 0)$$

Since the granularity of the approximation space defined by an atomic partition is the *finest*, the decision rules obtained from such a partition have the highest achievable certainty. These rules are summarized as follows:

$$r_{11} : DES(X_{11}) \xrightarrow{c_{11} = 1.0} nonrelevant$$

$$r_{12} : DES(X_{12}) \xrightarrow{c_{12} = 0.5} unknown$$

$$r_{13} : DES(X_{13}) \xrightarrow{c_{13} = 1.0} nonrelevant$$

$$r_{14} : DES(X_{14}) \xrightarrow{c_{14} = 0.5} unknown$$

$$r_{15} : DES(X_{15}) \xrightarrow{c_{15} = 1.0} nonrelevant$$

$$r_{16} : DES(X_{16}) \xrightarrow{c_{16} = 1.0} relevant$$

$$r_{17} : DES(X_{17}) \xrightarrow{c_{17} = 1.0} nonrelevant$$

Clearly, any approximation space defined by an atomic partition can easily produce a set of rules of unmanageable size in practical situations. More importantly, some index terms included in the atomic descriptions may be redundant and they unnecessarily make the decision rules less general. For example, we may consider another approximation space defined by the partition $\Pi_2 = \{X_{21}, X_{22}, X_{23}, X_{24}, X_{25}\}$ as shown below:

$$X_{21} = \{d_3, d_7, d_9\}, \quad X_{22} = \{d_8\}, \quad X_{23} = \{d_1\},$$
$$X_{24} = \{d_5, d_6\}, \quad X_{25} = \{d_2, d_4\},$$

and

$$DES(X_{21}) = (ALG \coloneqq 0)$$
$$DES(X_{22}) = (MAT \coloneqq 0) \wedge (PHY \coloneqq 0) \wedge (ALG \coloneqq 1)$$
$$DES(X_{23}) = (MAT \coloneqq 1) \wedge (PHY \coloneqq 1) \wedge (ALG \coloneqq 1)$$
$$DES(X_{24}) = (MAT \coloneqq 1) \wedge (PHY \coloneqq 0) \wedge (ALG \coloneqq 1)$$
$$DES(X_{25}) = (MAT \coloneqq 0) \wedge (PHY \coloneqq 1) \wedge (ALG \coloneqq 1)$$

The decision rules derived from such a partition are:

$$r_{21} : DES(X_{21}) \xrightarrow{c_{21} = 1.0} nonrelevant$$

$$r_{22} : DES(X_{22}) \xrightarrow{c_{22} = 1.0} relevant$$

$$r_{23} : DES(X_{23}) \xrightarrow{c_{23} = 1.0} nonrelevant$$

$$r_{24} : DES(X_{24}) \xrightarrow{c_{24} = 0.5} unknown$$

$$r_{25} : DES(X_{25}) \xrightarrow{c_{25} = 0.5} unknown$$

Note that the second set of rules is a generalization of the first one. For instance, the rule,

$$r_{21} = (ALG \coloneqq 0) \rightarrow nonrelevant ,$$

is a generalization of the following two rules:

$$r_{13} = (MAT \coloneqq 0) \wedge (PHY \coloneqq 0) \wedge (ALG \coloneqq 0) \rightarrow nonrelevant$$

$$r_{17} = (MAT \coloneqq 1) \wedge (PHY \coloneqq 0) \wedge (ALG \coloneqq 0) \rightarrow nonrelevant$$

It should be emphasized that the average certainty of the generalized rule is not affected by this generalization process. $\square$

We have developed a probabilistic learning algorithm (*PLA*) [16], capable of extracting a set of generalized rules from a given set of sample decisions in polynomial time. We believe that this algorithm provides a basic framework for designing a truly adaptive information retrieval system. In the next two sections, we will give a brief outline of the design methodologies of such a system.

6. AN ADAPTIVE INFORMATION RETRIEVAL SYSTEM

In information retrieval, we can perceive the user (inquirer) as an expert who knows how to distinguish relevant documents from nonrelevant ones, and regard the system as a *learner* which attempts to acquire the expert's knowledge such that it can correctly identify those relevant documents.

Let U denote the universe of all documents represented in the system. We can assume, without loss of generality, that all documents in U are indexed by a set of index terms. An index term t is said to be *sensitive* if there exists a possibility that those documents indexed by t might be relevant to the user. This initial guess of sensitive terms by the user prior to starting the learning session may considerably reduce the processing time. Initially, the system is presented with a sample $D \subseteq U$ of documents, which has been reviewed by the user. Thus, the relevance (nonrelevance) status of the individual document in D is known to the system. This sample of documents forms the basis of a knowledge representation system as described in Section 3.

Let R_p be the set of decision rules computed from the sample D, using the *PLA* algorithm. By the domain $D(r)$ of each decision rule $r \in R_p$, we mean the subset of those documents in the entire collection U, which satisfies the <condition> part of the rule r. We say that the set of decision rules R covers U if $U = \bigcup_{r \in R} D(r)$. That is, the relevance status of each document $d \in U$ can be determined by a certain rule from R. Since the *PLA* algorithm generates disjoint set of rules, for any document $d \in U$, there is in fact one and only one rule r in R which is applicable to d. Let $d \in U$ satisfies the rule $r = p \xrightarrow{c} q$ (i.e. d matches the description p of rule r), where p and q denote the <condition> and <conclusion> part of the rule respectively, and c is the certainty measure. The probability of relevance for a document d satisfying rule r (denoted by $P_{REL}(d)$) can be defined as follows:

$$P_{REL}(d) = \begin{cases} c & \text{if } q = \textit{relevant} \\ 1 - c & \text{if } q = \textit{nonrelevant or } q = \textit{unknown}. \end{cases}$$

The documents can then be ranked according to these probabilities of relevance as defined above, which is called the *rule-controlled* ranking. The retrieval process we suggest here essentially consists of two major steps. In the first step, the system generates an initial set of decision rules by asking the user to evaluate a number of sample documents. In the second step, the system provides the rule-controlled ranking of the documents. After reviewing the retrieved output, the user may decide to repeat these two steps in order to obtain a more refined set of rules if deemed necessary. This retrieval strategy can, perhaps, be explained more precisely in the following algorithm for an Adaptive Information Retrieval system (*AIR*).

7. THE *AIR* ALGORITHM

Let R_F denote the set of decision rules computed from a sample F of documents and let Z represent any subset of U.

Step 1. (generation of decision rules to cover U)

- 1.0 Let $Z = U$ and $F = \emptyset$.
- 1.1 Select a random sample E from Z and have the user evaluate all the documents in E.
- 1.2 Replace F by $F \cup E$.
- 1.3 Compute the set of decision rules R_F from the sample F (using the *PLA*).
- 1.4 Replace Z by $U - \bigcup_{r \in R_F} D(r)$.
- 1.5 Repeat step 1.1 to step 1.4 until $Z = \emptyset$.

Step 2. (rule refinement and document retrieval)

- 2.0 Let $Z = U - F$.
- 2.1 Repeat step 2.2 to step 2.6 until the user is satisfied or $Z = \emptyset$.
- 2.2 Generate the rule-controlled ranking of documents in Z using the rules of R_F.
- 2.3 Select a new sample E from the top ranked documents and have the user evaluate these documents.
- 2.4 Replace F by $F \cup E$.
- 2.5 Recompute the set of decision rules R_F from the sample F (using the *PLA*).
- 2.6 Replace Z by $U - \bigcup_{r \in R_F} D(r)$.

Example 6.

Suppose a knowledge database consists of the set of documents U given in Table 1. Assume that the user is interested in retrieving all those documents in U, which are indicated as being relevant by $REL = 1$. In the first step of the *AIR* algorithm, the user will be presented with several samples of documents for evaluation. Let the first sample be $E_1 = \{d_1, d_2\}$. By applying the algorithm *PLA* to this sample, we obtain the following decision rule:

$$r_{11} = (PHY \coloneqq 1) \xrightarrow{c_{11} = 1.0} \textit{nonrelevant}.$$

Note that this rule covers only the subset of documents $\{d_1, d_2, d_3, d_4\} \subset U$. This means that the next sample E_2 must be chosen from the set $U - \{d_1, d_2, d_3, d_4\} = \{d_5, d_6, d_7, d_8, d_9\}$. Let $E_2 = \{d_7, d_8\}$. By applying the *PLA* algorithm again to the expanded sample $E_1 \cup E_2 = \{d_1, d_2, d_7, d_8\}$, we obtain

$$r_{21} : (PHY \coloneqq 1) \xrightarrow{c_{21} = 1.0} \textit{nonrelevant},$$
$$r_{22} : (PHY \coloneqq 0) \wedge (ALG \coloneqq 0) \xrightarrow{c_{22} = 1.0} \textit{relevant},$$
$$r_{23} : (PHY \coloneqq 0) \wedge (ALG \coloneqq 1) \xrightarrow{c_{23} = 1.0} \textit{nonrelevant},$$

It can be easily verified that rules r_{21}, r_{22}, and r_{23} now completely cover U. At this point, step 1 of the *AIR* algorithm is completed.

Based on the rules obtained in Step 1, we can then rank the documents in $U - E_1 \cup E_2 = \{d_3, d_4, d_5, d_6, d_9\}$ according to their probabilities of relevance. (There is no need to consider $E_1 \cup E_2$ any more as this set has already been reviewed by the user.) Note that d_3 and d_6 are relevant with probability of relevance $P_{r_{22}}(d) = 1.0$ and d_5, d_4, d_9 are nonrelevant with $P_{r_{22}}(d) = 0$. Thus, the following ruled-controlled ranking of documents is produced: d_3, d_6, d_5, d_4, d_9. After reviewing the top ranked documents in the above list, the user may decide to continue the interaction with the system by inputing a larger sample in order to retrieve the remaining relevant documents in the collection. □

8. CONCLUSION

In this paper we have suggested a basic framework for the design of an adaptive information retrieval system. Our model is based on the notion of the probabilistic approximate classification we originally developed for learning by examples in artificial intelligence. The input to our algorithm is the probability distribution of some sample documents over a set of weighted index terms, which of course can be easily computed in practical applications. The decision rules obtained from our method are automatically generalized as only dominant and nonredundant index terms are selected in the construction of these rules. In a general sense, our method is related to the conventional probabilistic model. However, it must be noted that the problem of term independencies inherent in the probabilistic methods is absent in our approach.

For simplicity, we have assumed binary term weights in all the examples. This is, in fact, an unnecessary restriction as term weights other than 0 or 1 can be easily incorporated into our method. Furthermore, we can also deal with *relevance grading* as in the theory of fuzzy sets [23] by replacing the binary relevance status *(yes* or *no)* with some other finer degrees of relevance factors if so desired.

We have already successfully built an experimental *AIR* system based the ideas presented here. The preliminary results obtained so far are very encouraging and detailed evalation of the model is in progress.

REFERENCES

[1] K. Sparck Jones, Automatic Keyword Classification for Information Retrieval, Butterworths, London (1971)

[2] G. Salton and M. J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, New York (1983)

[3] G. Salton, The SMART Retrival System - Experiment in Automatic Document Processing, Prentice-Hall, Englewood Cliffs, New Jersey (1971)

[4] S. K. M. Wong, W. Ziarko, P. C. N. Wong, 'Generalized vector space model in information retrieval', Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, (1985), 18-25

[5] T. Radecki, 'Fuzzy set theoretical approach to document retrieval', Information Processing and Management, 15 (1979), 247-259

[6] A. Bookstein, 'Fuzzy Requests: An approach to weighted Boolean searches', Journal of the American Society for Information Science, 13 (1980), 240-247

[7] R. M. Tong, V. N. Askman, J. F. Cunningham, and C. J. Tollander, 'RUBRIC - an environment for full text information retrieval', Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, (1985), 243-251

[8] J. J. Rocchio, 'Relevance feedback in information retrieval', The SMART Retrieval System - Experiment in Automatic Document Processing, Prentice-Hall, Englewood Cliffs, New Jersey (1971) (edited by G. Salton)

[9] C. J. van Rijsbergen, Information Retrieval, Butterworths, London (1980)

[10] C. T. Yu and G. Salton, 'Precision Weighting - An effective automatic indexing method', Journal of the ACM, 23 (1976), 273-286

[11] C. T. Yu, W. S. Luk, and T. Y. Cheung, 'A statistical model for relevance feedback in information retrieval', Journal of the ACM, 23 (1976) 273-286

[12] S. E. Robertson and K. Sparck Jones, 'Relevance weighting of search terms', Journal of the American Society for Information Science, 27 (1976), 129-146

[13] C. J. van Rijsbergen, 'A theoretical basis for the use of co-occurrence data in information retrieval', Journal of Documentation, 33 (1977), 106-119

[14] A. Bookstein and D. Kraft, 'Operations research applied to document indexing and retrieval decisions', Journal of the ACM, 24 (1977), 410-427

[15] S. E. Robertson, 'Theories and models in information retrieval', Journal of Documentation, 33 (1977), 126-148

[16] S. K. M. Wong and W. Ziarko, 'INFER--An adaptive decision support system based on the probabilistic approximate classification' Proceedings of the 6th International Workshop on Expert Systems and Their Applications, Avignon, France, 1(1986), 713-726

[17] R. S. Michalski, 'Theory and methodology of inductive learning', Machine Learning : The Artificial Intelligence Approach (eds. R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Tioga Press, Palo Alto (1983))

[18] Z. Pawlak, 'Rough sets', International Journal of Information and Computer Science, 11 (1982), 341-356

[19] Z. Pawlak, 'Rough classification', International Journal of Man-Machine Studies, 20 (1984), 469-483

[20] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Machine Learning : The Artificial Intelligence Approach, Tioga Press, Palo Alto (1983)

[21] J. R. Quinlan, 'Learning efficient classification procedures and their application to chess end games', Machine Learning : The Artificial Intelligence Approach (eds. R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Tioga Press, Palo Alto (1983))

[22] C. E. Shannon, 'A mathematical theory of communication', Bell Systems Technical Journal, 4 (1948), 379-423

[23] L. A. Zadeh, 'Fuzzy sets', Information Control, 8 (1965), 338-353

AN AUTOMATIC AND TUNABLE DOCUMENT INDEXING SYSTEM

Esen OZKARAHAN
Dept. of Computer Science
Arizona State University
Tempe, Arizona
USA, 87287

Fazli CAN
Dept. of Electrical
& Electronics Engineering
Middle East Technical Univ.
Ankara, Turkey

ABSTRACT

In this article we present an interactive automatic document indexing software together with various index tuning/optimization strategies. After stems are generated from the raw text, the initial index vocabulary is narrowed down and tuned with the use of indexing versus clustering theory relationships. The narrowed down vocabulary is further optimized with the inclusion of term phrases and virtual terms corresponding to high and low frequency terms respectively. The results of performance experimentation which proved significant improvements of index vocabulary optimization are presented. The exploitation of the term discrimination value concept in index and retrieval system tuning and optimization is discussed.

1. INTRODUCTION

In this article we will present an automatic and tunable document indexing system which is used in the integrated document and database retrieval system of the RAP GDBMS (Generalized Database Management System) [OZKA86]. Index tuning is accomplished by the methodologies based on the cover coefficient concept [CAN85a] which was developed in document clustering.

Indexing in document retrieval is used to represent documents by structures that use index terms. In this way, the representation is accomplished in a much compact form. The structures that utilize index terms can be inverted files or document vectors. Information retrieval can be achieved either entirely by the use of indexing or by using indexing for database partitioning and then searching partitions by full text search [HOLL84]. In RAP GDBMS, indexing is used to construct document vectors which are used for document clustering to partition the database. Partitions are then searched by full text search. Automatic indexing refers to determining the document representatives (i.e., the index terms) from within documents by some computerized means. Alternative to automatic indexing is manual or controlled indexing which imposes upon documents certain index terms which represent a specific area of the discipline that the documents are believed to belong. Such methodology, of course, lacks generality in terms of retrieval and true representation of documents.

In this article we will present an experimental automatic indexing system. Contrary to manual indexing automatic indexing brings with it the analytical power of indexing theory. And one useful aspect of this power is the ability to tune various parameters of the retrieval system. Such tuning provides effectiveness for the retrieval system and its performance, and efficiency for the underlying computer system.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

2. THE AUTOMATIC INDEXING SYSTEM

2.1 System Structure

Figure 1 summarizes the overall structure of the indexing system in terms of the basic building blocks and currently operational software. Input to the system is raw documents (text) in computer readable form and output consists of document vectors structured into a document by term matrix which is referred to as the D matrix. According to this figure, the system is broken into two major parts: basic automatic indexing and tuning & optimization.

2.2 Indexing Programs

In this part of the indexing system there is always need to download and/or upload a tree in alphabetical order to save time for various processing. And at the same time the tree structures must be balanced for search efficiency. To create a balanced binary tree from a sorted input the program ***balance*** contains an insertion scheme where, with the size of the list known, the input list is halved recursively and inserted into the tree to create a balanced tree.

At the beginning, the sorted list of stop words is input to the program ***doc_stop*** and converted to a balanced tree called the ***common_tree***. Next the random (unsorted) stream of raw text (document) files are read and the ***common_tree*** is searched. The ***doc_stop*** program builds a binary tree (reasonably balanced with random input) from those document file terms that do not find a match in the ***common_tree***. The resulting structure is called the ***start_list***. Both the ***start_list*** and raw text contain zone delimiters for proper weighting that takes place subsequently.

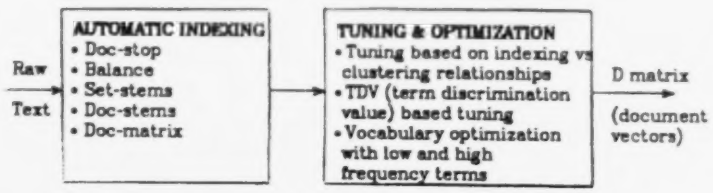


Figure 1. Overall structure of the indexing system

The (English) natural language introduces certain problems to be dealt with. They are special spellings, acronyms, synonyms, prefixes, word endings etc. which must be handled carefully and accurately. A fully automatic and suitable stem generator is hard to achieve. Even the best known systems are known to be about 70% efficient. There are several reasons for this. First of all there is the well known morphological problems of the English language (e.g., absorb vs absorption; mouse vs mice). Then comes the problem of how to treat the stemmed adjectives adverbs etc. For example, what should we do with the words such as "red", "friendly". While most adjectives may not be important as an index term some others may be required as in the case of "very large databases". The next problem is in the selection of full or subset of technical word stems. All these points helped us recognize the difficulty of achieving a "full blown" automatic indexing software. An interactive system with user interaction, such as the system described here, was deemed essential.

The next thing after building the *start_list* tree is to download it in sorted order and identify stem groups. A stem group is the group of terms which cluster in the sorted list and share the shortest stem of the group. However, stemming is not an easy task. We cannot simply pick the shortest stem as the stem of the group.

Consider, for example, the terms ***adding***, ***adds***, ***added***, or ***addition***, ***additional***, and ***additionally***. A problem arises in stemming the first three words to "add". Should the term "add" be matched with the term "additional" or even the term "address"? If all three of these terms were stemmed too far in the *start_list*, the queries would be very inaccurate. Let us look at the following section of the *start_list* that was produced from our experimental TODS database:

add	added	adding
adds	addition	additional
additionally	address	addressed
addresses	addressing	

As can be seen, all of these words have the common "add" as the first three letters and this is a stem in itself, therefore, these words form a stem group. A rule of thumb is to chop off the common endings as far as possible, yet leave enough of the stems to differentiate between other stems in the group. For example all these terms would stem down to:

add
addition
address

Next is to insure that the shortest stem is at the beginning of the stem group and the remaining stems within the group are in alphabetical order. Using this rule of thumb the initial *start_list* for the TODS database was reduced from over 3100 words down to 1532 stemmed terms called the *stem_list*. This is a significant decrease in storage and increase in speed without losing accuracy when searching and matching stemmed terms. With the *start_list* in alphabetical order, it is easy with any screen editor to determine the groups of stems and chop off endings accordingly.

Before explaining how to use the program that handles this newly edited *stem_list*, here is another example of stemming a group of terms in the *start_list*. First is the alphabetical list of the terms as they appeared in the *start_list*:

reliability	reliable	rely
relate	related	relating
relation-based	relational	relations
relationship	relationships	relative
relatively	release	releasing
relevance	relevant	relieve
relieving	relnet	relocate
relocating	relocation	

All these terms begin with the three letters "rel" and contain many common endings. After chopping off the common endings, insuring the shortest stem is first, and insuring the remaining stems in the group are in alphabetical order, the resulting group of stems include:

rel
relat
relation
relative
releas
relevan
rellev
relnet
relocat

To complete the editing session, the *start_list* must be edited to produce a *stem_list* with 1) the shortest stem in the group first and this stem is referred to as the base stem, 2) the remaining stems in the group in alphabetical order, and 3) all stems free of common endings.

The *start_list* after stemming is called a *stem_list*. The *set_stems* program prepares this newly edited *stem_list* for insertion into a tree called the *stem_tree*. *set_stems* maneuvers the stems into the correct order to insure a) the resulting tree will be balanced, and b) each group of stems will remain intact for accurate frequency and weight counting.

First the base stem of each stem group is removed from the list. The balancing algorithm uses the resulting list of base stems to build a balanced tree. *set_stems* places the remaining stems alphabetically in a file at the end of the balanced base stem list. The base stems become the tree nodes in the resulting tree called *stem_tree*. To keep the stem groups together in a linked list structure connected to the appropriate tree node, *set_stems* takes the remaining stems of the sorted

list and searches the *stem_tree*. Whenever a base stem node is matched the stem is added to the linked list emanating from that node.

The program *doc_stems* is the most important link in the chain of programs to generate index vocabularies and D-matrices. The balanced *stem_list* in the form of *stem_tree* and the raw text (document) file are the inputs of the *doc_stop* program. *doc_stop* searches *stem_tree* with the raw text input. This is done because an input term may not find a match in *stem_tree* since 1- it is a stop word, 2- it is a discarded term such as an adjective, name, or a totally insignificant or irrelevant term, and 3- several terms may converge on a common stem (i.e., terms of the stem group) so that correct frequencies and weights can be obtained. In searching *stem_tree*, if a term finds a match with a node the search follows a linked list attached to the node containing the stem group. This insures that the correct stem of the group is matched. The frequency and weight of each term are cumulatively added to the correct stem. Frequency is simply the total number of occurrences of a stem. Weight is the sum of frequency times the zone weight of a stem over all zones and documents. Assuming the *stem_list* was edited correctly, the tree would have been built correctly so that the appropriate stems will reflect the correct weight and frequency values. For example the word *relational* will first find a match at the *stem_tree* node with the shortest stem in the group "rel". Then the algorithm follows down the linked list holding the stem group and adds the frequency and appropriate weight to the stem *relation* and not the stem *rel*. Likewise the word "reliable" will match at the node with "rel", traverse down the linked list and not find another appropriate stem. The frequency and weight will be added to the correct stem "rel". The recursive code that handles this unique algorithm is found in the program *doc_stem*.

The stemmed terms with their corresponding frequencies and weights computed are used to produce information for the remaining programs and cluster algorithms. If a list of terms within certain frequency and weight bounds is desired, the *w* menu option must be used. This list is called an index vocabulary and used to generate vectors for the clustering algorithm. The program queries the user for the desired min and max weights and frequencies and also the output file for the results. All stemmed terms within those weight and frequency bounds are generated in alphabetical order.

Two convenient menu options are *d* (for down) and *u* (for up) for downloading the tree information into a nontext file of records and uploading these records back into the tree. As indicated at the beginning, this saves time by reloading the tree without rescanning the document file. These options were used when generating multiple vocabularies and testing for the best seed vectors for clusters. Furthermore the records can be uploaded using a different key. For example if the tree was downloaded in alphabetical order, it can be uploaded in weight or frequency order. The TODS *stem_list* was downloaded and uploaded in weight order. This was used to produce the terms with the highest to lowest weights.

With each stemmed term an array of boolean (i.e. a binary vector), as long as the total number of documents, is kept. This array indicates which documents the particular term is found in. The array becomes part of the D-matrix required for the program *doc_matrix* and the clustering algorithms.

The *doc_matrix* program is a utility that provides all the stems for a given document or all the documents a given stem has occurred in. This program also computes the quantity t which is the total usage of the index vocabulary. This is computed from the nonzero entries in the D-matrix.

3. INDEX TUNING AND OPTIMIZATION

3.1 Choosing An Index Vocabulary

Choosing an index vocabulary constitutes the rough tuning of our system. The initial number of terms (stems) resulting from *doc_stems* is usually more than needed for an adequate vocabulary. The "adequacy" of a vocabulary is the subject of (or determined by) our tuning methodology. There are certain theoretical and experimental relationships which we can use at the start. In the experiments of [CAN85b] index terms were generated by using certain frequency and weight range thresholds conjunctively (e.g., $8 \leq$ weight ≤ 45 AND $5 \leq$ frequency ≤ 15). This resulted in index vocabularies of various sizes.

In the current system the index vocabulary is narrowed down to a range that presents a vocabulary smaller than the *doc_stems* output but somewhat larger than the final form. To do this we vary the conjunctive frequency and weight range thresholds in such a way that the range is spread around the $\log_2 m$ value. This value is the expected number of documents per cluster should the documents of the database be clustered [SALT75a], m being the total number of documents. In actuality one need not cluster documents, however, the use of clustering theory would verify the $\log_2 m$ value for a given index vocabulary. In our research this aspect has paramount significance. As reported in [CAN85ab] we have observed very important relationships between indexing and cover coefficient based clustering. This means that even though one chooses not to cluster documents the clustering methodology can be helpful in producing adequate index vocabularies. To recapulate the relationships observed between indexing and the cover coefficient based clustering the following subsection is included.

3.2 Indexing Versus Clustering Relationships

The cover coefficient based clustering methodology implies some relationships between indexing and clustering. These relationships are the following:

$$n_c = \frac{t}{z_d \times t_g} \qquad d_c = m/n_c = t_g$$

$$n_c = \frac{m \times n}{t} \leq \frac{m}{t_g} \leq \frac{n}{z_d} \qquad d'_c = n/n_c = z_d$$

where,

- n_c: number of clusters implied by the cover coefficient concept;
- t: $\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}$, i.e., the total number of term assignments in the binary D matrix;
- z_d: average indexing exhaustivity, i.e., the average number of terms used for the description of a document (t/m);
- t_g: average term generality, i.e., the average number of documents which use a term (t/n);
- m: number of documents;
- n: number of terms;
- d_c: average number of documents in each docu-

ment cluster;

d'_e: average number of terms in each term cluster.

In [CAN85a], these relationships are also shown analytically by using the formula given for the number of clusters based on the cover coefficient concept. For the reader who is unfamiliar with the cover coefficient concept [CAN84,CAN85a] the appendix provides a brief introduction.

In [CAN85b] these relationships were experimentally observed by using 14 different D matrices generated from the TODS database. (At that time this database contained 167 articles, however, currently there are 214 articles). Table 6 in [CAN85b] summarized these relationships. In that table it can be observed that $n_c \approx \frac{m \times n}{t}$ (where $n_c = \sum_{i=1}^{m=167} \delta_i$), $d_c \approx t_g$, and $d'_c \approx z_d$. We have observed the validity of these relationships also in hierarchical clustering. In hierarchical clustering document description vectors are used at the lowest level. At the higher levels centroid vectors are used. The hierarchical clustering methodology and the centroid generation policy can be found in [CAN84,OZKA84]. The indexing experiments for hierarchical clustering, which were not presented in [CAN85b], are shown in Table 1. The initial index vocabulary was generated using the T4 experiment of Table 2 in [CAN85b].
Table 1 can be examined for the validity of the relationships indicated in the foregoing.

In terms of index tuning, we can see that once we have a fix on certain quantities the other unknown quantities can be determined from the relationships. For example, if we know our m, d_c (initially $\log_2 m$) then we can determine n_c which in turn leads to n via $n_c = \frac{n}{z_d}$ where $z_d = d'_c = \log_2 n$.
Table 2 presents a series of experiments performed [BECK85] for narrowing down the index vocabulary. This table also presents the other quantities so that indexing versus clustering relationships can be observed.

Referring to Table 2, for 214 documents ($=m$), $d_c = t_g = 7.743$ is taken as the starting point of the index vocabulary range. This corresponds to $\log_2 214$. *doc_stem* produced index terms (stems) sorted with respect to the frequency & weight values. The frequency and weight ranges shown in Table 2 correspond to the range of what we call the "goodterms". These ranges are chosen such that the middle frequency terms with high resolution are covered for the index vocabulary. Once an index vocabulary was determined, the clustering software was run to generate clusters. After observing document distributions in the resulting clusters it was seen that the index vocabulary (corresponding to the second to last row of Table 2) with 551 terms gave the best (uniform) distribution with 24 clusters and hence was chosen for the retrieval experiments. The rest of the article will be devoted to 1) index vocabulary optimization with respect to low and high frequency terms and 2) the use of cover coefficient based term discrimination value (TDV) in index tuning in conjunction with clustering. Even though we are currently experimenting with (1) and (2) within their own merits, an "all-connected" automatic indexing and tuning software is not yet achieved. When this can be done, the n number of index terms can be determined with the help of a list of terms sorted on TDVs, rather than merely using the middle range of the frequency & weight based list.

3.3 Optimizing The Vocabulary With High & Low Frequency Terms

In [CAN85a] we introduced, for the first time, the idea of optimizing an index vocabulary with the use of a cover coefficient based methodology and presented two algorithms. The algorithms were on the treatment of high and low frequency terms for index optimization. At this time we are in a position to present the results of our experimentation on the matter. Before we discuss the experiments we will first restate the algorithms in the following. Again, for a reader unfamiliar with the cover coefficient concept we suggest reading of the appendix. The reader should note that in the following we will be referring to term clustering which is discussed in the latter part of the appendix.

In indexing, it is known that medium frequency terms are usually good discriminators, whereas the terms at the extremes, that is, the

Table 1. Experiments in hierarchical clustering

cluster level	m	n	t	$(m \times n)/t$ (rounded)	n_c (rounded)	t_g	d_c	z_d	d'_c
1	167	539	4185	22	22	7.73	7.59	24.94	24.50
2	22	459	1099	9	10	2.39	2.44	49.95	51.00
3	10	198	325	6	5	1.64	2.00	32.50	39.60
4	5	47	79	3	3	1.68	1.67	15.80	15.67
5	3	18	26	2	2	1.44	1.50	8.67	9.00

Table 2. Index vocabulary experiments

Frequency Min	Frequency Max	Weight Min	Weight Max	# of Stems (Vocabulary) (n)	Total in D-matrix (t)	# of docs per stem (t_g)	#of clusters expected $((214 \times n)/t)$
5	15	8	45	253	1736	6.86	31.18
5	20	7	55	358	2624	7.329	29.19
4	25	7	60	401	3157	7.873	27.18
6	20	9	55	274	2224	8.114	26.36
7	20	10	55	235	2042	8.689	24.62
7	25	10	60	268	2546	9.50	22.52
4	41	4	79	551	4995	9.065	23.61
10	59	10	116	309	4568	14.783	14.48

high frequency and low frequency terms are poor contributors to precision and recall. To improve precision and recall, usually the high frequency terms are replaced by contiguous word phrases and low frequency terms are replaced by concept hierarchies.

With respect to our terminology, if t_i and t_j are two high frequency terms, then t_{ij} $t_{ij} \gg t_g$. In other words, their term generality will be much greater than the average term generality. In this case, instead of using t_i and t_j, we would rather use a new term t_{ij} if t_i and t_j exhibit considerable association between them. We can measure such an association very easily by using our coupling (decoupling) coefficients.

In the case of low frequency terms, we can cluster these terms and map each term into a virtual term in indexing and/or searching processes.

3.3.1 Treatment of High Frequency Terms

a) Include the high frequency terms in the index vocabulary.

b) Compute the seed power p'_i of index terms where $p'_i = \delta'_i \psi_i t_{gi}$, and t_{gi} is the generality of term-i.

c) Sort the term seed powers in descending order. In this way very frequent terms will appear at the beginning of the sorted list.

d) Determine a cut off for the high frequency terms.

e) The terms which satisfy the following conditions can be used for constructing the composite terms: to construct a phrase t_{ij} from two terms t_i and t_j $p'_i \approx p'_j$, $\delta'_i \approx \delta'_j$, $c'_{ij} \approx c'_{jj}$ should hold. And the mutual coverage of the terms (c'_{ij} and c'_{jj}) should have a value comparable with $\delta'_i(\delta'_j)$.

Notice that step (b) will put the terms with the same or close term generality consecutively into the resorted list. If these terms satisfy the conditions set forth in step (e), then this means that the terms have the same pattern of distribution among documents. If c'_{ij}, c'_{jj} have a value comparable with $\delta'_i(\delta'_j)$, then this means that the terms are assigned to the same set of documents. Therefore, they can be used for constructing a composite term t_{ij}. In finalizing the formation of a composite term, which by the way is identical to a word phrase, user judgment is necessary. This is because we cannot commute the order of terms in the natural language and achieve the same meaning. The concept of composite formation can be applied to n number of terms, where $n \geq 2$.

3.3.2 Treatment of Low Frequency Terms

We can propose a similar methodology for improving the indexing quality of the low frequency terms. The following is a procedure to achieve this:

a) Include the low frequency terms in the index vocabulary.

b) Compute the term seed powers as in step (b) of high frequency terms.

c) Sort the term seed powers in descending order. The very infrequent terms will appear at the bottom of the list.

d) Determine a low frequency term cut off.

e) Construct low frequency term clusters by use of the C' matrix and the single pass term clustering algorithm.

f) For each low frequency term cluster (i.e., for the terms of the cluster and centroid) a virtual term is assigned. That is, all terms of each cluster are represented by (or mapped to) the virtual term. In our experiments, the virtual term is simply the cluster number.

In an information retrieval system, any index or query term that belongs to the infrequent term category must be mapped to the virtual term of its cluster either manually or automatically in a way similar to the way a concept hierarchy is taken from a thesaurus.

3.3.3 Experimentation

In our experimental system we tested the foregoing concepts [ALGH86]. The results are rather encouraging. Our experimental TODS database is the same as outlined in [CAN85ab]. The differences are the new stemming procedure outlined at the beginning and the increased number of documents. The total number of documents is now 214 where each document contains the title, abstract, and the keywords of a TODS article. The following summarizes the parameters of the experimentation.

Total number of documents	214
Number of terms in the index vocabulary	634
Number of clusters	22
Average number of documents per cluster	9.727
Average number of terms in a centroid	153.2
Number of queries	19
Average number of terms in a query	19.05
Average number of significant terms in a query	7.53

The index vocabulary of the previous section, which consisted of "goodterms" only was extended with the low and high frequency terms, referred to as the "low" and "high" terms respectively. This resulted in a total of 1532 index terms of which 551 were goodterms, 95 high frequency terms, and 886 low frequency terms. After this, high frequency term phrases and term clusters (for virtual terms) were generated. The following list shows the twenty four high frequency term phrases that were constructed.

concurren access
concurren control
control structur
database design
database manag distribut process
database manag system
database system
distribut concurren control
distribut database
distribut database manag system
distribut database system
function depend
implement technique
quer process
quer record
relation database
relation database schema
relation database system
relation model
relation operat
relation schema
retriev updat operat
statistic database
system perform

Also, 59 term clusters were constructed from the low frequency terms. About sixteen percent of the resulting term clusters were discarded due to their still insignificant frequency (e.g., ≤ 3). This

meant that of the total of 886 low frequency terms 748 were included in the chosen clusters to correspond to virtual terms. The following is a sample low frequency cluster corresponding to virtual term # 2.

cluster 2
total freq. = 13
18 members
portion,compos,data-swap,move,back,return,job,
restart,rollback,invariant,spread,deadly,embrace,
prevent,terminat,abort,morphism,bring

The 214 documents of the database were reclustered using the new index vocabulary that contained the virtual terms and high frequency term phrases. As shown in the table at the beginning of the section, this reclustering with the new index vocabulary of 634 terms resulted in 22 clusters. The entire database was classified into 22 subtopics (coincidence, unrelated to the number of clusters). A query had at least two or more relevant documents of a subtopic. Then three target clusters [CAN85a] were determined by a search of document vectors of the entire database and sorting documents with respect to their similarities with a given query. Documents in the target clusters were then manually examined for relevance. Nineteen queries were used to generate average recall and precision measures. Experiments were run for different combinations of the index vocabulary that consisted of a) goodterms only, b) goodterms with high term phrases, and c) the virtual terms added to (b), the resulting set is referred to as "all terms". Three matching functions were used. They are 1) Dice's coefficient, 2) Cosine similarity and 3) The Coupling Function we have introduced in [CAN85a] which is based on the cover coefficient concept and considers the term distributions besides the query, document vector pair. Figure 2 shows the comparisons of the matching functions with respect to the 19 queries when documents represented with the goodterms only. Figure 3 shows the same, only this time both the high term phrases and low term clusters (virtual terms) are incorporated with the goodterms. Figure 4 shows a comparison of the three combinations using a single matching function.

In these experiments we can observe that the inclusion of the high frequency terms in the form of term phrases improves recall and precision. When both the virtual terms and the high frequency term phrases were incorporated the overall performance of the information retrieval system was improved considerably, especially in precision.

4. USE OF TERM DISCRIMINATION VALUE (TDV) IN INDEX TUNING

4.1 The TDV Concept

In a document space, Term Discrimination Value (TDV) is used to measure the effect of an index term on the distinguishability of documents from each other. The distinguishability (or separation) of documents can be measured by looking at the document space density (i.e., the average similarity of documents), Q' or Q which are

$$Q' = \frac{2}{m(m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} S(d_i, d_j) \quad (0 \le Q' \le 1, \text{ for } 0 \le S \le 1)$$

or

$$Q = \frac{1}{m} \sum_{i=1}^{m} S(d_i, G) \quad (0 < Q \le 1 \text{ for } 0 < S \le 1)$$

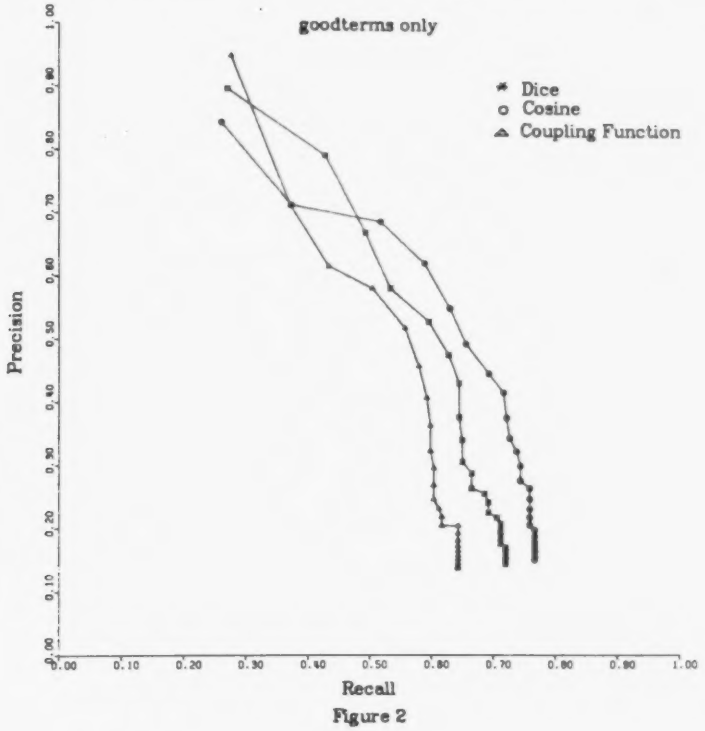


Figure 2

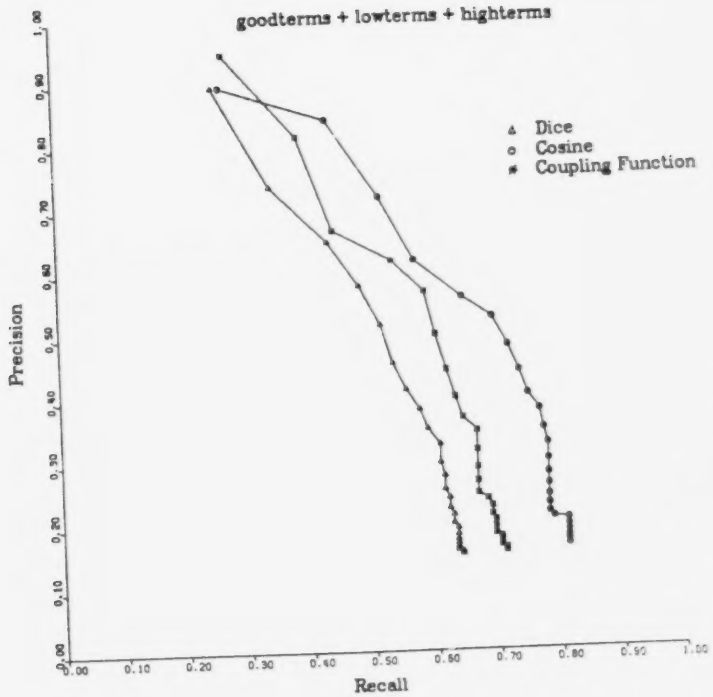


Figure 3

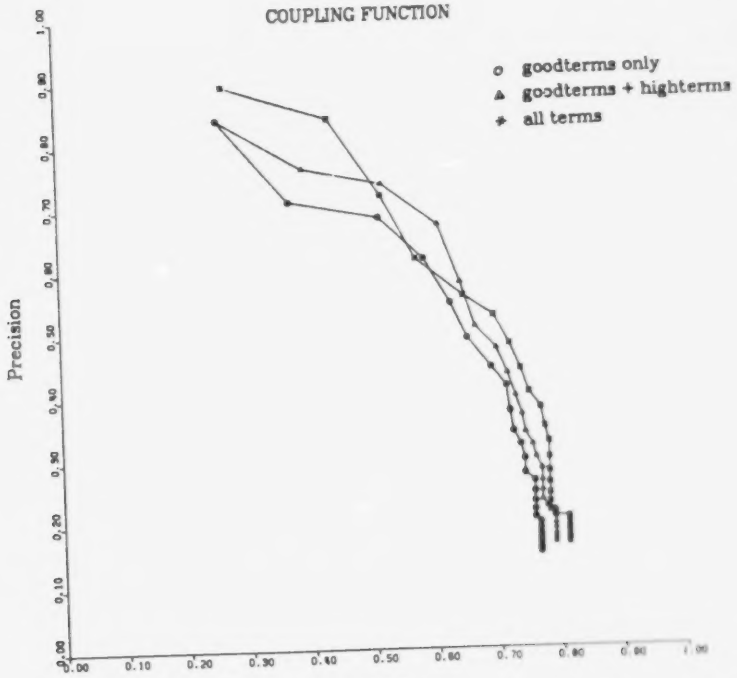


Figure 4

where S and G denote a similarity function and a document centroid, respectively.

Let T be the set of index terms to be used to describe a document collection. Deletion of a term from T will change the indexing vocabulary and hence the description of documents and of course the document space density Q. The deletion of t_j ($1 \le j \le n$) will set $d_{ij}=0$ ($1 \le i \le m$) and hence $g_j=0$. The new value of Q, Q_j, will be the following:

$$Q_j = \frac{1}{m}\sum_{i=1}^{m} S(d_i^j \cdot G_j)$$

where

$d_i^j=(d_{i1},d_{i2},\ldots,d_{i,j-1},d_{i,j+1},\ldots,d_{in})$, $G_j=(g_1,g_2,\ldots,g_{j-1},g_{j+1},\ldots,g_n)$

(Notice that, in this case $T_1=T$ and $T_2=T-\{t_j\}$ where T_1 and T_2 are the original and modified indexing vocabularies, respectively.)

The difference $(Q_j - Q)$ reflects the change due to deletion of term t_j. If the assignment of t_j makes the documents more separated (closer) from (to) each other then its assignment will decrease (increase) the document space density and hence $Q<Q_j$ ($Q>Q_j$). Accordingly, (Q_j-Q) will be greater (less) than 0. The difference (Q_j-Q) is defined as the discrimination value of t_j, TDV_j [SALT75ab,SALT83]. Accordingly, for a better description of documents we should use index terms that have high, positive TDVs.

We can exploit the concept of TDV with the use of the cover coefficient related variables. If we consider the notions of document (term) coupling and decoupling we can realize that the concepts of document space density (Q) and average decoupling of documents (δ or n_c) are inverse to each other. Table 3 shows the interpretation of the related quantities with respect to TDV (where δ and δ_i are the average decoupling of documents before and after the deletion of t_i).

Table 3. Effects of the type of index term (t_i) on the values of Q,δ (or n_c)

	Quantity	
Type of t_i	Q vs Q_i	δ vs δ_i (or n_c vs n_{ci})
Good discriminator ($TDV_i>0$)	$Q<Q_i$	$\delta>\delta_i$ ($n_c>n_{ci}$)
Indifferent discriminator ($TDV_i\approx0$)	$Q\approx Q_i$	$\delta\approx\delta_i$ ($n_c\approx n_{ci}$)
Poor discriminator ($TDV_i<0$)	$Q>Q_i$	$\delta<\delta_i$ ($n_c<n_{ci}$)

According to the cover coefficient concept, TDV_i is defined as (n_c-n_{ci}), i.e., the change in the number of clusters after the deletion of t_i. In [CAN86] it is shown that the (exact) term discrimination value according to the cover coefficient concept can be computed as follows:

$$TDV_i = \sum_{i=1}^{f_i} \left| \delta_i - \alpha_i^j \times \left[\frac{\delta_i}{\alpha_i} - d_{ij}^2 \times \beta_i \right] \right|$$

where $f_i=|D_i|$ (i.e., cardinality of D_i) and $D_i=\{\, d_i \mid d_i \in D \text{ and } d_{ij}\neq 0\}$, and $\alpha_i^j=(\alpha_i^{-1}-d_{ij})^{-1}$ (i.e., the reciprocal of the sum of row-i of the D matrix excluding d_{ij}). For a binary D matrix $d_{ij}^2=d_{ij}$ and $\alpha_i^{-1}-(\alpha_i^j)^{-1}=1$ if $d_{ij}=1$, for $1\le i\le m$, $1\le j\le n$. Hence, for a binary D matrix, the above formula will take the following form:

$$TDV_i = \sum_{i=1}^{f_i} \alpha_i^j \times (\beta_i - \delta_i)$$

It should be mentioned that the TDV values according to the cover coefficient concept are exact values, since their calculation does not involve any approximation (e.g., use of centroids).

4.2 System Tuning with TDVs

In designing and implementing an IR system we may want to vary various parameters. If we are using clustering we may need to tune cluster size due to external constraints. In the cover coefficient based clustering we can predict the cluster size analytically. However, for better precision or recall the designer (administrator) may want to decrease or increase cluster size. Also, for efficiency of the underlying computer system we may want to optimize cluster (partition) size with respect to paging and memory constraints. In hierarchical clustering, we may also want to control the number of clusters at each level of the hierarchy. In binary indexing, the relationships, we discussed at the beginning, between indexing and clustering can be used to accomplish this. For example, if higher recall is important we can keep term generality, t_g, high, hence also the cluster size since it was observed that $t_g \approx d_c$.

In general, however, we may want to control cluster size in any retrieval system. The tuning concept based on TDV can be used for this purpose. The administrator of an IR system may specify the number of clusters that he (she) wants to have within a collection. Assume that N_c is the number of clusters that the administrator wants to have within a collection. (Remember that, the number of clusters implied by the cover coefficient concept is within the range of 1 to min (m,n); therefore, N_c should be within this range.) The selection of index terms can be done according to N_c. That is, one could choose the terms to be used for indexing by observing the effect of the selected terms on the number of clusters.

Let n_c be the number of clusters implied by the original D matrix either by the cover coefficient clustering or by $(m/\log_2 m)$ [SALT75a]. The comparison of n_c with N_c would indicate one of the following term selection methodologies:

a) $N_c \approx n_c$: the current definition of the D matrix satisfies the administrator's requirement. In this case no change in D is required.

b) $N_c > n_c$: the current number of clusters implied by the D matrix is smaller than the administrator's requirement. In this case the terms which lessen the number of clusters should be deleted from T. (The set of terms used for the description of documents.) These are terms which have negative TDVs.

c) $N_c < n_c$: the current number of clusters implied by the D matrix is greater than the administrator's requirement. In this case the terms which increase the number of clusters should be deleted from T. These are terms which have positive TDVs.

A solution to these procedures is presented in [CAN86] which also presents the computation of term/document discrimination values with the use of the cover coefficient concept. A detailed analysis of the computational complexity is also included.

5. CONCLUSION

In this article we presented the automatic and tunable indexing system which is used in the integrated database and document retrieval system called RAP GDBMS. The interactive software accepts raw text and produces an initial set of index terms after removing stop words and stemming the remaining text words. This initial set of terms is narrowed down and tuned by the use of frequency and weight thresholds in conjunction with certain relationships we derived between indexing and the cover coefficient based clustering.

The narrowed down vocabulary is further optimized by incorporating term phrases and virtual terms derived from high and low frequency index terms, respectively. For this purpose, term phrasing and term clustering algorithms which are based on the cover coefficient concept are utilized. The optimized index vocabulary is compared with the version before optimization by a set of retrieval experiments. The experiments clearly showed that significant performance improvements can be obtained in terms of recall and precision. And this is possible by automatic tuning and optimization.

In the concluding part of the article, we briefly discussed exploitation of the term discrimination value concept in index tuning and optimization. The cover coefficient version of the TDV computation is now available with its accompanying methodology for system tuning.

REFERENCES

[ALGH86]
ALGHAMDI, J.S. *Implementing and Testing the Cover Coefficient Based Information Retrieval System*, MSc Thesis, Dept. of Computer Science, Arizona State University, Tempe, Arizona, 1986.

[BECK85]
BECKER, A.L. *Testing Database Software and Implementing Information Retrieval Research*, MSc Thesis, Dept. of Computer Science, Arizona State University, Tempe, Arizona, 1985.

[CAN84]
CAN, F., OZKARAHAN, E.A. *Two Partitioning Type Clustering Algorithms*. **Journal of the American Society for Information Science**. 35(5): 268-276; 1984.

[CAN85a]
CAN, F., OZKARAHAN, E.A. *Concepts of the Cover Coefficient-Based Clustering Methodology*, **Proceedings of ACM SIGIR Conference**. June 1985. Montreal, Canada: 204-211.

[CAN85b]
CAN, F., OZKARAHAN, E.A. *Similarity and Stability Analysis of the Two Partitioning Type Clustering Algorithms*. **Journal of the American Society for Information Science**. 36(1): 3-14; 1985.

[CAN85c]
CAN, F. *A New Clustering Scheme for Information Retrieval Systems Incorporating the Support of a Database Machine*. Ph.D. dissertation, Middle East Technical University, Ankara, January, 1985.

[CAN86]
CAN, F., OZKARAHAN, E.A. *Computation of Term/Document Discrimination Values by Use of the Cover Coefficient Concept*. **Journal of the American Society for Information Science**, to appear.

[HOLL84]
HOLLAAR, L. *The UTAH Text Retrieval Project - A Status Report*, **Proceedings of ACM Conference on Research and Development in Information Retrieval**. July 1984. Cambridge, England. 123-132.

[OZKA84]
OZKARAHAN, E.A., CAN, F. *An Integrated Fact/Document Information System for Office Automation*, **Information Technology: Research and Development**. 3(3):142-156, 1984.

[OZKA86]
OZKARAHAN, E. **Database Machines and Database Management**, Englewood Cliffs, New Jersey: Prentice-Hall; 1986.

[SALT75a]
SALTON, G. **Dynamic Information and Library Processing**. Englewood Cliffs, New Jersey: Prentice Hall; 1975.

[SALT75b]
SALTON, G. *A Theory of Indexing*. **Regional Conference Series in Applied Mathematics** No.18, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania; 1975.

[SALT83]
SALTON, G., McGILL, M.J., **Introduction to Modern Information Retrieval**, New York: McGraw Hill, 1983.

ACKNOWLEDGMENT

The authors gratefully acknowledge the contributions of Jarallah Alghamdi, Alan M. Becker, and Gregory Hardyman of Arizona State University in implementing and testing various indexing, clustering, and information retrieval software.

APPENDIX
The Cover Coefficient Concept

The cover coefficient (CC) concept was originally introduced for document clustering purposes [CAN84,CAN85abc,OZKA84]. The document collection is defined by the D matrix which is already described. The entries of the D matrix, d_{ij} ($1 \le i \le m$, $1 \le j \le n$) satisfy the following conditions:

[a] $\sum_{j=1}^{n} d_{ij} > 0,\ 1 \le i \le m$ (each document has at least one term).

[b] $\sum_{i=1}^{m} d_{ij} > 0,\ 1 \le j \le n$ (each term is assigned to at least one document).

The D matrix is mapped onto an $m \times m$ C (cover coefficient) matrix. Each c_{ij} entry of the C matrix indicates the extent with which $\text{document}_i(d_i)$ is covered by $\text{document}_j(d_j)$.

The C matrix is formed by using the matrices S and S' as $C = S \times S'^T$. Entries of the S and S' matrices are defined as follows:

$$s_{ij} = d_{ij} / \left(\sum_{k=1}^{n} d_{ik}\right), \quad s'_{ij} = d_{ij} / \left(\sum_{k=1}^{m} d_{kj}\right) \qquad 1 \le i \le m, 1 \le j \le n$$

s_{ij} (s'_{ij}) indicates the significance of $t_j(d_i)$ for $d_i(t_j)$. The entries of the C matrix are defined as follows:

$$c_{ij} = \sum_{k=1}^{n} s_{ik} \times s'^{T}_{kj}$$

$$= \sum_{k=1}^{n} (\text{significance of } t_k \text{ in } d_i) \times (\text{significance of } d_j \text{ for } t_k)$$

where S'^T indicates the transpose of the matrix S'

From the definition of the S and S' matrices

$$c_{ij}=\alpha_i\times\sum_{k=1}^{n} d_{ik}\times\beta_k\times d_{jk} \quad (1\le i\le m,\ 1\le j\le m)$$

where α_i and β_k are the reciprocal of the row_i and $column_k$ sums, respectively.

Each entry of C is a covering coefficient among documents and $\sum_{j=1}^{m} c_{ij}=1$ for $1\le i\le m$. The diagonal entries c_{ii} $(1\le i\le m)$ of the C matrix indicate the extent with which d_i is covered by itself and called the uniqueness or decoupling coefficient, δ_i, of d_i. $c_{ik}\ge c_{ij}$ if $i\ne j$ for a binary D matrix, however this condition does not hold for a weighted D matrix [CAN84]. This is very simple to prove. Consider c_{ik} and c_{ij}:

$$c_{ik}=\alpha_i\times\sum_{k=1}^{n} d_{ik}^2\times\beta_k$$

$$c_{ij}=\alpha_i\times\sum_{k=1}^{n} d_{ik}\times d_{jk}\times\beta_k$$

If $d_{ik}\ge d_{jk}$, and if $d_{ik}>d_{jk}$ for at least one k $(1\le k\le n)$, then $c_{ij}<c_{ik}$.

If none of the terms of d_i is used by any other document then $\delta_i = 1$ (i.e., d_i is completely unique or "decoupled" from the other documents of the collection), otherwise $\delta_i < 1$ meaning that d_i is coupled with one or more documents of the collection. Thus the values of δ_i fall in the range $0<\delta_i\le 1$.

The sum of the off-diagonal entries of row_i is referred to as the coupling coefficient ψ_i of d_i. ψ_i indicates the extent of coupling of d_i with the other documents in the collection. $\psi_i=1-\delta_i$ and from the definition of δ_i, the value range of ψ_i is $0\le\psi_i<1$ [CAN84]. The overall or average decoupling coefficient of the collection is

$$\delta=\sum_{i=1}^{m}\delta_i/m=trace(C)/m \quad (0<\delta\le 1)$$

It is hypothesized that the number of clusters within a collection, n_c, is $\delta\times m$ = trace(C). The value range implied for n_c by δ is $1\le n_c\le \min(m,n)$ [CAN85ac]. The average number of documents within a cluster would be $m/n_c=m/(\delta\times m)=1/\delta$. The concept of decoupling coefficient implies that increase in δ or in individual δ_i $(1\le i\le m)$ will increase the number of clusters $(n_c=\delta\times m=\sum_{i=1}^{m}\delta_i)$ and hence decrease the average size of clusters. The use of such cover coefficient related concepts for clustering purposes is illustrated in [CAN84,CAN85c].

Estimation of the number of clusters within a collection is one of the unresolved problems of clustering. The computation of n_c in the CC based methodology can be considered an heuristic. Currently, we are working to show the validity of $n_c=\delta\times m$ in the CC based clustering methodology.

A similar $n\times n$ matrix called C', which has properties similar to the C matrix, can be constructed for index terms. C' is defined as the product $S'^T\times S$. The individual entries of the C' matrix are defined as follows:

$$c'_{ij}=\sum_{k=1}^{m} s'_{ki}\times s_{kj} = \sum_{k=1}^{m} (\text{significance of } d_j \text{ for } t_k)\times(\text{significance of } t_k \text{ in } d_i)$$

From the definition of S and S' matrices

$$c'_{ij}=\beta_i\sum_{k=1}^{m} d_{ki}\times\alpha_k\times d_{kj} \quad (1\le i\le n,\ 1\le j\le n)$$

The concepts of decoupling and coupling coefficients of $term_j$, respectively, δ'_j and ψ'_j are the counterparts of the same concepts defined for documents. Hence $\delta'_j=c'_{jj}$, and $\psi'_j=1-\delta'_j$. The concepts of overall coupling and decoupling coefficients are also defined for terms and represented by δ' and ψ', respectively.

The number of clusters implied by δ' is $n'_c=\delta'\times n$. It is shown that $n'_c=n_c$ [CAN85c]. The C' matrix can be used for constructing clusters of index terms and more effective indexing vocabularies. The diagonal entries of the C' matrix are used for the construction of the centroid vectors for document clusters and in the calculation of seed powers, $p_i(1\le i\le m)$ of individual documents. The cluster seed power is determined as:

$$(p_i=\delta_i\times\psi_i\times\sum_{j=1}^{n} d_{ij} \text{ and } p_i=\delta_i\times\psi_i\times\sum_{j=1}^{n}(d_{ij}\times\delta'_j\times\psi'_j)$$

in the case of binary and weighted D matrices, respectively. The C matrix and the documents which have the highest cluster seed power are then used in the implementation of the two partitioning type clustering algorithms.

Performance of Self-Taught Documents: Exploiting Co-Relevance Structure in a Document Collection.*

Abraham Bookstein
Graduate Library School
University of Chicago

Abstract

In this paper we study the behavior of an information retrieval system in which index terms are assigned at random to both documents and requests. The random indexing is then modified by means of a feedback mechanism derived from a normal probability model and applied to both the request and document representations. Of interest is the convergence properties of the representation vectors. After few feedback iterations, it is found that well defined clusters form that accurately represent the co-relevance structure among the documents—in effect the feedback mechanism has permitted the documents to index themselves. This approach offers an interesting way to extend the dimensionality of the indexing vocabulary. Both this application and a theoretical analysis of the impact of extending the indexing vocabulary are discussed.

1. Introduction

The size of its index vocabulary is one of the factors that determines how well an Information Retrieval System (IRS) can perform. This size imposes, for example, an intrinsic limit on the number of relevance classes that the system can recognize—with binary indexing, an IRS using n terms cannot distinguish more than 2^n documents, or more than 2^{2^n} concepts, if a concept is defined as a Boolean expression based on the index vocabulary. If the collection includes more than 2^{2^n} concepts, then, clearly, some Boolean expressions must denote more than a single concept, and even with perfect indexing and perfect request formulation, an IRS cannot perfectly match requests and documents (the consequences of this restriction have been noted by Radecki (1985)). In practice, this theoretical limit becomes significant because most of concept space, as represented by binary document vectors, is sparsely populated, while some regions are dense with documents; in such regions, retrieval would benefit from the ability to subdivide larger classes. Further, in a crowded space, classes are closer together, so errors in the indexing of documents and requests are more likely to result in the retrieval of inappropriate documents. What is desired is the addition of index terms that can separate classes of documents with indentical profiles into subclasses based on differences in the concepts included in the class, and that also can, in the coding theory sense (Patterson and Weldon (1972)), increase the average distance between classes of documents, reducing the system's sensitivity to error. For example, five additional index terms would permit a single class to be divided into 32 subclasses, allowing us to distinguish components of a set of documents brought together by their indexing. The problem is how to expand effectively the index space, and apply productively these terms to a set of documents. (The impact of additional index terms on retrieval performance is discussed further in the Appendix.)

The purpose of this paper is to suggest such a technique. Our approach is based on random indexing followed by a learning process. Specifically, we are suggesting that the dimensionality of the concept space as represented by the index terms can be artificially extended by a small number of pseudo-index-terms. The weights assigned to these terms in both the documents and requests are initially generated randomly, and then modified with experience through a learning process—both for the request and the documents. It is our hope that, over time, the additional index terms will permit an evolution of classes of documents into subclasses in a manner that conforms with the relevance structure of documents implied by the requests presented to the IRS. To study how documents behave in such a process, in this paper we shall consider the extreme case of purely random indexing and concentrate on how the random component by itself evolves under learning.

The goal of document retrieval is to present to a user a set of documents that has a higher utility, or lower cost, than any other set of documents. The fundamental assumption that we are making, and that is implicit in all feedback systems, is that the probability distribution over the sets of documents, indicating the probability that a particular set will be the optimal one to retrieve, is sharply peaked, and that documents tend to co-occur in the same optimal sets. Thus, documents are linked together by their tendency to be relevant for the same requests, and these links imply a co-relevance structure that is invisible before the system garners retrieval experience with the collection, but which is exploitable once such experience is available. One function of indexing is to create approximate co-relevance classes of documents, bringing together documents that have been similarly indexed. In this paper we are exploring the possibility that the latent co-relevance structure itself is adequate, through the feedback process, for creating classes of documents useful for retrieval. We shall also comment on some implications that this process has for indexing in general.

* This material is based upon work supported by the National Science Foundation under Grant IST-8413559.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

2. The Learning Process

The detailed learning process we shall use is motivated by and extends the sequential learning model as described by Bookstein (1983a). That model assumes a fixed document space and an imperfect request. On the basis of the request, documents are retrieved in sequence and presented to the user. Under the assumption that terms, or more generally, document features, have different distributional characteristics among documents relevant and not relevant to a request, and using standard Bayesian manipulation techniques, algorithms are presented for modifying the request—specifically the weights assigned to index terms in the request. These modified weights are then used to retrieve additional documents.

In the specific model described below, documents and requests appear symmetrically. A natural extension of the model, then, is to consider the documents as well as the requests to be imperfectly indexed, and to benefit by retrieval experience. Just as currently each request "sees" a stream of documents, each of which is evaluated and helps modify the request; similarly, in our extension of this process, each document sees a stream of requests, to some of which it is judged relevant, to others not relevant. Formally, this is a process very similar to that by which the requests are evaluated, and we are suggesting that the documents be modified in a similar manner as well. The full learning process can then be described as follows:

When a request is presented, provide the user with a stream of documents. As these documents are evaluated, modify the request to improve subsequent retrievals. As this process continues, the request more perfectly represents the intent of the user initiating the search than does the initial query, given the current indexing of the document collection. Finally, given this educated query, teach the document set (or at least those evaluated) to be more responsive to that query and those like it. This process is similar to that of Brauen (1971), but formulated within, and motivated by, a probabilistic retrieval framework.

Given the above processing, it is interesting to ask about the role of the initial indexing. In principle, even with random indexing, the document vectors may well evolve to form useful patterns of term weights, thereby indexing themselves on the basis of feedback information alone. From this point of view, the retrieval process is conceptualized as a learning process in which both requests and document representations evolve to form optimal, complementary patterns in a manner similar to the genetic algorithms described by Gordon (1985); the purpose of the initial indexing is to establish locations on this trajectory reasonably close to the ultimate optimal resting place, and thus is essentially an efficiency mechanism. We shall refer below to the sequential process without initial indexing as a "*pure* learning process." Although we are not suggesting here that documents and requests in fact be indexed at random, as noted above, exactly such a pure learning process may be useful for guiding the pseudo-terms by which the index vocabulary is extended.

Since we are concentrating on the pseudo-terms alone, we are, in effect, studying the pure learning process defined in the preceeding paragraph. The questions we would like to answer are:

[1] Does the process converge?

[2] Do the random vectors evolve toward useful patterns?

[3] How fast is the evolution?

[4] How are the terminal patterns of document indexing affected by various relevance patterns among the documents?

[5] How does the size of the index vocabulary (in practice, the vocabulary of the pseudo-terms) influence performance?

The purpose of this paper is to emphasize the potential usefulness of extending the index vocabulary, and to suggest a mechanism by which this can be accomplished. Below we shall describe the results of some early experiments on this problem.

3. Retrieval Model

As a first attempt to model a pure learning process, we shall adopt a simplified version of the normal term independence model defined in Bookstein (1983b). In that model both documents and requests are represented by vectors of weights. It is assumed that the relevance weights for each term are normally distributed in both the sets of relevant and nonrelevant documents, though with different mean values. Since the modification process is additive, we expect that even if the weights are not normally distributed initially, they will be more so after several iterations.

The retrieval weight, W, for a document derived from the normal model is as follows:

$$W = \sum_i d_i[(\mu_i/\sigma_i^2 - \bar{\mu}_i/\bar{\sigma}_i^2) - d_i(1/\sigma_i^2 - 1/\bar{\sigma}_i^2)/2] \qquad 1)$$

where d_i is the weight of the i-th term for the document and $\mu_i(\sigma_i)$ and $\bar{\mu}_i(\bar{\sigma}_i)$ are the means (standard deviations) for the term among relevant and nonrelevant documents, respectively. We shall simplify the formula by assuming $\sigma_i = \bar{\sigma}_i$, and that the variation of σ_i among terms is negligible. Thus, to within a constant factor, the weight can be restated as

$$1a)\quad W = \sum d_i(\mu_i - \bar{\mu}_i).$$

Here, the fixed vector $\mathbf{d} = \{d_i\}$, is the document representation. The vector $(\mu_i - \bar{\mu}_i)$ represents the request during the current iteration—it indicates how important term i is in the request by measuring the differences between the weight the term takes among relevant documents as contrasted with its weight among non-relevant documents. Thus, the weights comprising the request vector are derived from the retrieval experience and approach as a limiting value a vector, $\mathbf{r}$, as more of the document set is evaluated. In this paper, we shall evaluate the request on all the documents, and after this evaluation, the limiting value $(\mu_i - \bar{\mu}_i)$ will form the vector $\mathbf{r}$ which will represent the request during the next stage of the process. The retrieval weight at this stage, denoted by W, is the inner product of $\mathbf{d}$ and $\mathbf{r}$. Once an optimal $\mathbf{r}$ is found for each request, symmetry suggests that the document vectors be reassessed as $d_i = r_i - \bar{r}_i$; that is, the document weight for the i-th term is the difference between the weights of that term in requests for which the document is relevant and those for which it is non-relevant.

Users arrive at an IRS with diverse interests and desiring a variety of document packages. A statement of exactly which documents a user desires is the most perfect form a request can take. Ideally, we would like to represent both documents and requests in a manner such that both effectively cooperate to generate such a set in a given instance. The indexing of documents and the formulation of requests are mechanisms to approximate such a representation. We are describing an

iterative process that is intended to approach more perfectly those ideal representations. This interpretation of information retrieval as a dual process in the course of which both the document indexing and the request seek their optimal values has been presented before by Bookstein (1985).

4. Methodology

Our simulation, programmed in C, is driven by three matrices:

a. A matrix of document-request relevance patterns, created artificially to examine how such patterns affect retrieval performance—it is here that the document co-relevance structure is introduced into the model;

b. A document-term matrix providing document indexing information; and

c. A parallel matrix of request-term information.

The information provided in b) and c) is strongly influenced by the parameterization suggested in Bookstein (1983a). For example, the request matrix will include, for each term, the following values:

a. the number of relevant documents evaluated;

b. the sum of the weights for that term over those relevant documents;

c. the number of non-relevant documents evaluated;

d. the sum of weights for that term over those non-relevant documents; and

e. the retrieval weight as suggested by eqn. 1a).

Various initial values of parameters a) and c) will be tested, and initial values for b), d), and thus e), will be generated at random. These values will then be modified in the course of the iterative retrieval process, with the value for e) being used for the retrieval itself. In practice, the values a) and c) are "confidence" measures, since the larger they are, the more documents must be evaluated before the retrieval weight, e), is significantly affected. For random assignments, a small value will be used.

In this model, documents and requests are conjugate entities, and the data structure representing the document set is similarly constructed.

5. Results

In this preliminary investigation, we are particularly interested in the following question: Will like documents and requests successfully cluster? That is, does random indexing generate, through statistical fluctuation, patterns that can be amplified and exploited through feedback?

To test these we used a set of four documents and four requests randomly indexed by ten pseudo-terms (one might think of the four documents as a quartet not readily distinguishable on the basis of their manually assigned index values). We imposed a relevance structure in which the first two documents were relevant to the first two requests, and the second pair of documents was relevant to the second pair of requests. We then systematically ran all the requests through all the documents. The initial indexing as shown in Table 1, reveals no obvious pattern associating, for example, the first request with either the second request or either of the first two documents. This observation is supported by Table 2, which shows the retrieval weights for the documents (rows) for each request (column). For example, the first request would retrieve the first document (correctly) and last (incorrectly) with positive weights, and assigns negative weights to the second and third document.

Initial Request Matrix				Initial Document Matrix			
requests				documents			
1.0	-1.0	-4.0	-1.0	1.0	-1.0	0.0	0.0
1.0	2.0	0.0	0.0	0.0	0.0	-2.0	0.0
-1.0	2.0	1.0	-1.0	0.0	-1.0	-1.0	-1.0
-1.0	1.0	1.0	0.0	-1.0	1.0	2.0	-1.0
0.0	0.0	-1.0	2.0	-1.0	0.0	1.0	1.0
0.0	-1.0	-2.0	0.0	0.0	1.0	-1.0	-1.0
1.0	1.0	-2.0	1.0	1.0	0.0	1.0	0.0
-1.0	1.0	0.0	2.0	-2.0	0.0	1.0	0.0
-1.0	0.0	0.0	1.0	0.0	2.0	3.0	-2.0
-2.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0

TABLE 1. Tables of Random Indexing. Each matrix gives the initial, randomly assigned weights indicating the strength with which each of ten pseudo-index terms is attached to each of four documents or requests. Each column constitutes a random index record. The vertical bar divides the documents and requests into equi-relevance classes, so, for example, the requests on the left of the bar are relevant to the documents on the left of the bar in the document matrix.

	Requests			
Documents	5.0	-3.0	-6.0	-6.0
	-3.0	-1.0	2.0	4.0
	-8.0	0.0	1.0	10.0
	2.0	-1.0	0.0	2.0

TABLE 2. Initial matrix of retrieval weights. Each weight represents the strength with which a document is retrieved for a request before feedback. The main diagonal quadrants are of relevant document-request pairs. The minor diagonal is of non-relevant document-request pairs.

Tables 3 and 4 show the corresponding results after three of the four requests have been evaluated. Patterns are now indeed emerging. For example, examining the index records, we find requests one and two agree regarding which terms are assigned positive or negative weights, and tend to differ in this regard from requests three and four. A similar statement can be made for the relationship between requests and documents. This pattern is revealed most strongly by the matrix of retrieval weights, which in all cases correctly relate documents to requests in agreement with our relevance assumptions. It seems clear that patterns occurring in random indexing can be profitably exploited, and surprisingly quickly. It also suggests that feedback mechanisms can be very effective in quickly correcting errors and misjudgements that may occur in human indexing.

Final Request Matrix				Final Document Matrix			
requests				documents			
0.3	-0.3	-1.1	-1.0	0.9	0.1	-0.6	-0.6
1.0	1.5	-1.1	0.0	1.2	1.4	-2.1	-1.4
0.0	0.2	1.0	-1.0	-0.6	-1.1	0.3	0.1
-0.7	-0.8	1.8	0.0	-1.9	-1.1	2.4	1.0
-1.0	-2.2	2.7	2.0	-2.9	-2.6	3.3	2.9
1.0	0.7	-1.5	0.0	1.1	1.6	-1.6	-1.6
0.3	0.1	-0.2	1.0	0.5	0.0	0.5	-0.2
-1.3	-1.9	2.9	2.0	-3.7	-2.5	3.0	2.7
0.0	-0.7	1.3	1.0	-1.0	-0.2	2.4	0.4
-1.3	-1.9	3.3	1.0	-2.9	-3.1	3.2	3.4

TABLE 3. Index table after iterating over three of the four documents. The format is the same as in Table 1.

	Requests			
Documents	15.8	23.2	-37.7	-16.9
	13.8	20.1	-31.9	-12.4
	-16.9	-26.2	41.2	18.9
	-15.0	-21.9	34.2	15.2

TABLE 4. Retrieval weights after iterating over three of the four documents.

6. Conclusions

The immediate purpose of this paper is to test a mechanism for index space expansion that uses random indexing and feedback. More generally, this paper is a comment on the character of indexing itself—it indirectly asks, why and under what conditions is indexing an effective mechanism for preparing a collection of documents for retrieval. This is basically a psychological question, but it has formal implications. In this simulation, after several iterations, the pseudo-terms in effect index the document collection. In this case, the indexing represents the document collection's co-relevance structure.

The retrieval problem can be viewed as follows: Given a set of N documents, which of the 2^N possible subsets that one might in principle retrieve is the optimal one?

From this perspective, indexing becomes in effect a code for representing sets of documents. Since each possible Boolean expression corresponds to a set of documents, it is possible to represent 2^{2^n} sets of documents with a set of n index terms, or, equivalently, it is possible to distinguish 2^n documents with n index terms. If each subset of documents is equally likely to be the optimal one for a random request, no simplification is suggested by coding theoretic arguments. However, because of document co-relevance, as well as the costs associated with retrieving too few or too many documents, some subsets of documents have a much smaller probability of being optimal than others. Coding theoretic arguments suggest that this structure should permit us to reduce the dimensionality of our index space so as to better correspond to the Shannon Entropy of the power set of documents (though this may require us to coalesce sets of documents very unlikely to be optimal). Thus, the Shannon Entropy forms a type of lower bound on the dimensionality of the index space. In this paper, documents index themselves on the basis of feedback information. If this could be done in an actual retrieval environment, the dimensionality of the space of the resulting index vectors, determined, perhaps, through factor analytic techniques (Deerwester), would provide an estimate of this lower bound.

Appendix: Effects of increasing concept space dimensionality

In the text we noted the impact of introducing new index terms (or pseudo-terms) on our ability to discriminate among subclasses of a class of documents, and on diminishing the effect of errors in indexing, both precision increasing mechanisms. We shall here attempt a more precise statement of these effects.

1) Increased discrimination

Additional index terms permit us to construct a larger number of classes, as noted in the text. This is clearly the case with binary indexing over a finite vocabulary, where the total number of possible classes can be computed explicitly. We shall now extend this argument to the case in which the indexing weights are taken from a more general, and possibly infinite, set of values. We shall assume that different possible document vectors have different probabilities of occurring.

Consider a set of optimally indexed documents. Since the greater the effective number of document classses generated by the indexing the greater the uncertainty as to which class will next be chosen, an inviting measure of discrimination power here is the Shannon entropy (Ash, 1985). If each document is represented by an n-dimensional index vector, and $f(x)$ is the probability density of that vector taking the value x, then the entropy H is defined by

$$H=-\sum_x f(x)\log f(x).$$

Suppose we now add the term x_{n+1} in a manner that doesn't change the (now) marginal distribution of x; call the expanded $n+1$ dimensional vector x'. Using the relation $f(x')=f(x)f(x_{n+1}|x)$, with $f(x_{n+1}|x)$ denoting the probability that the $n+1$-st term, x_{n+1}, take a particular value given the values of the original index weights, x, we find:

$$H'=\sum f(x')\log f(x')=-\sum f(x')\log f(x)-\sum f(x')\log f(x_{n+1}|x)$$

The first term on the right hand side can be summed over x_{n+1}, reducing $f(x')$ to the marginal distribution $f(x)$, which we are assuming to be equal to the initial distribution. Thus the first term is the initial entropy, H.

The second term, upon expanding x', becomes $-\sum f(x)f(x_{n+1}|x)\log f(x_{n+1}|x)=\sum f(x)H_{n+1}(x)$, where $H_{n+1}(x)$ denotes the entropy of the term x_{n+1}, conditional on the value of x; since $f(x_{n+1}|x)$ is a legitimate probability distribution, this is a non-negative quantity. The second term, then, is the expected value of this conditional entropy and is also non-negative; it is sometimes called the conditional uncertainty of the variable x_{n+1} given the variable x (Ash, 1965).

In the above sense, adding additional terms to an index vocabulary expands the number of classes available for distinguishing classes of documents. Our argument suggests a precise measure of this discrimination ability, and indicates how an additional index term affects this measure. It also underscores the importance of using index terms that occur in a moderate number of documents and are independent of other index terms, as these increase the functions $f(x_{n+1}|x)$ and $H_{n+1}(x)$.

2) Error resilience

By increasing the space available for document vectors, increasing index space dimensionality reduces the likelihood that a small indexing error will put a document into a different retrieval class. To make this observation more precise we will use two probability distributions.

a. f(x): probability that a request (assumed to be accurately indexed for this argument) is represented by a point, x, in concept space; and

b. g(x): probability that a document *not* about x is indexed as x, for x an n-dimensional vector.

Below, the subscripts n and $n+1$ on a function will denote the dimensionality of its argument.

If, given the request x' we retrieve documents in a small volume, V, around x', then $e_n(x') = \int_V g_n(x)dx$ denotes the probability of an erroneous retrieval to request x', and $e_n = \int f_n(x)e_n(x)dx$, integrated over the full n-dimensional space of index values, the likelihood of a false retrieval. Suppose we now extend the concept space by appending to the vector, x, a term x_{n+1}. As in part 1, we assume that the distribution of the initial index terms is unchanged. Let V_{n+1} denote a small interval around x_{n+1}; below, x and x', without subscripts, will denote full $n+1$ dimensional vectors, or the first n components of such a vector, depending on the subscript on f or g.

Then $e_{n+1}(x') = \int_{V_{n+1}} dx_{n+1} \int_V dx g_{n+1}(x)$. Noting that

$\int_{-\infty}^{\infty} dx_{n+1} g_{n+1}(x) = g_n(x)$ and that

$\int_{-\infty}^{\infty} dx_{n+1} \int_V dx g_{n+1}(x) \geq \int_{V_{n+1}} dx_{n+1} \int_V dx g_{n+1}(x)$, we conclude that

$e_n(x) \geq e_{n+1}(x)$. Thus

$e_{n+1} = \int f_{n+1}(x) e_{n+1}(x) dx \leq \int f_{n+1}(x) e_n(x) dx$,

and, explicitly integrating over x_{n+1} on the right hand side, we conclude $e_{n+1} \leq e_n$.

References

[1] Ash, R. 1965. *Information Theory*. New York, NY: Interscience Publishers Division of John Wiley and Sons, 1965. 335 pp.

[2] Bookstein, A. 1983a. "Information Retrieval: A Sequential Learning Process." *Journal of the American Society for Information Science* (1983), vol 34, no. 5: Pp. 331-342.

[3] Bookstein, A. 1983b. "Explanation and Generalization of Vector Models in Information Retrieval." In: Salton, G., and Schneider, H-J. (eds), *Research and Development in Information Retrieval: Proceedings of the Conference: 1982 May 18-20; Berlin, West Germany*. Berlin, West Germany: Springer-Verlag, 1983. Pp. 118-132.

[4] Bookstein, A. 1985. "Representing Boolean Structure within a Probabilistic Framework." In: *Recherche d'Informations Assistée par Ordinateur. Proceedings of the International Symposium: 1985 March 18-20: Grenoble, France*. Grenoble, France: I.M.A.G., 1985. Pp 373-386.

[5] Brauen, T.L. 1971. "Document Vector Modification." In: Salton, G. (ed), *The SMART Retrieval System: Experiments in Automatic Document Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1971. Pp. 450-484.

[6] Gordon, M.D. 1985. "A Learning Algorithm Applied to Document Redescription." In: *Research and Development in Information Retrieval*. Eighth Annual International ACM SIGIR Conference, 1985 June 5-7; Montreal, Quebec. New York: ACM, 1985. Pp. 179-185.

[7] Peterson, W.W., and Weldon, E.J. 1971. *Error-Correcting Codes*. Cambridge, Mass: MIT Press, 1971.

[8] Redecki, T. 1985. "Output Ranking Methodology for Document-Clustering-Based Boolean Retrieval System." In: *Research and Development in Informatin Retrieval. Eighth Annual International ACM SIGIR Conference*. 1985 June 5-7; Montreal, Quebec. New York: ACM, 1985. Pp. 70-76.

[9] Deerwester, Scott. Work in progress

Two Models of Retrieval with Probabilistic Indexing

Norbert Fuhr
TH Darmstadt, FB 20 FG DVSII, D-6100 Darmstadt

Abstract

We describe two retrieval models for probabilistic indexing. The binary independence indexing (BII) model is a generalised version of the Maron & Kuhns indexing model. In this model, the indexing weight of a descriptor in a document is an estimate of the probability of relevance of this document with respect to queries using this descriptor. The retrieval-with-probabilistic-indexing (RPI) model is suited to different kinds of probabilistic indexing. Therefore we assume that each indexing model has its own concept of 'correctness' to which the probabilities relate. The concept of correctness is not necessarily identical with the concept of relevance, it is only required to depend on relevance. In addition to the probabilistic indexing weights, the RPI model provides the possibility of relevance weighting of search terms. Both retrieval models are compared in experiments, showing equally good results.

1. Probabilistic indexing models

The first paper on probabilistic indexing was published by Maron & Kuhns ([MARO60]) 26 years ago. The central idea of this model is to estimate for each descriptor in a document a probability of relevance - the probability that the document is relevant to a request which is formulated with this descriptor. But this model has never been investigated in experiments, because of the problem of estimating the required probabilistic parameters. All suggestions to solve this problem (see also [MARO82]) require too much intellectual effort.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.
© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

In the meantime, other models of probabilistic, automatic indexing have been developed which are based on certain forms of document representation. The well-known 2-Poisson Model ([HART75], [BOOK74,75]) uses the within-document frequency of terms for the estimation of the indexing weights. The Darmstadt indexing approach (DIA), which has been developed at the TH Darmstadt, West Germany, since 1978, is based on a more detailed document representation (see chapter 3). In the past, the probabilistic parameters of this model have been estimated by comparison with manual indexing. A similar model was suggested in [ROBE84].

In this paper, a new model for retrieval with probabilistic indexing is proposed which is suited for different kinds of probabilistic indexing. Especially the probabilistic parameters derived from manual indexing can be given an interpretation in this model and can be related to the probability of relevance. The second model presented here is a generalisation of the Maron & Kuhns model. All models are tested in experiments, using the DIA for the estimation of the required probabilistic parameters.

2. The binary independence indexing model

The binary independence indexing (BII) model, which will be described here, is a generalised version of the Maron & Kuhns model. Every specific user request to a retrieval system must be transformed into a query with descriptors from the set $S = \{s_1, \ldots, s_n\}$. We assume that every query can be represented as a binary vector $\vec{x} = (x_1, \ldots, x_n)$ with

$$x_i = \begin{cases} 1, & \text{if the query contains descriptor } s_i \\ 0, & \text{otherwise.} \end{cases}$$

In this way, a specific user request f_k is mapped onto a vector $\vec{x}_k$ (different requests may have the same query).

The event space of the BII model consists of all document-request relationships between the set of all documents in the collection and all requests to the system. As the set of all requests is not completely known, we assume that we have knowledge about a representative sample of it. A document-request relationship is either relevant or non-relevant, which will be denoted by R and $\bar{R}$.

The BII model seeks for an estimation of $P(R|\vec{x}_k, d_m)$, the probability that the document d_m is relevant to a request using query $\vec{x}_k$. Now three independence assumptions are possible:

$$P(\vec{x}_k) = \prod_{i=1}^{n} P(x_{k_i}) \qquad (I1)$$

$$P(\vec{x}_k|R, d_m) = \prod_{i=1}^{n} P(x_{k_i}|R, d_m) \qquad (I2)$$

$$P(\vec{x}_k|\bar{R}, d_m) = \prod_{i=1}^{n} P(x_{k_i}|\bar{R}, d_m) \qquad (I3)$$

All three assumptions relate to the distribution of descriptors in the queries. $I1$ says that the distribution of the descriptors in all queries is independent, whereas $I2(I3)$ says that the distribution of the descriptors is independent only in those queries where the document d_m is relevant (non-relevant) to the corresponding request.

Using assumptions $I1$ and $I2$, we get the ranking formula

$$P(R|\vec{x}_k, d_m) = P(R|d_m) \cdot \prod_{i=1}^{n} \frac{P(R|x_{k_i}, d_m)}{P(R|d_m)} \qquad (BII1)$$

Here $P(R|d_m)$ denotes the probability that d_m is relevant to an arbitrary request, $P(R|x_{k_i}, d_m)$ is the probability that document d_m is relevant to an arbitrary request which contains descriptor s_i in its query (x_{k_i}=1) resp. where s_i is not present in the query (x_{k_i}=0).

With $I3$ instead of $I1$, we get an odds formula, where $O(X) = P(X)/P(\bar{X})$:

$$O(R|\vec{x}_k, d_m) = O(R|d_m) \cdot \prod_{i=1}^{n} \frac{O(R|x_{k_i}, d_m)}{O(R|d_m)} \qquad (BII2)$$

In the original model of Maron & Kuhns, the following simplifying assumption was made implicitely:

$$\prod_{\substack{i \\ x_{k_i}=0}} P(x_{k_i}=0) \approx \prod_{\substack{i \\ x_{k_i}=0}} P(x_{k_i}=0|R, d_m) \qquad (V1)$$

From this follows that the relevance of a document with respect to a request depends only on those descriptors which are present in the query, and not on those descriptors which the query does not contain.

With $I1, I2$ and $V1$ we get the original ranking formula of Maron & Kuhns:

$$P(R|\vec{x}_k, d_m) = P(R|d_m) \cdot \prod_{\substack{i \\ x_{k_i}=1}} \frac{P(R|x_{k_i}=1, d_m)}{P(R|d_m)} \qquad (BII3)$$

Finally replacing $I1$ by $I3$, the ranking formula yields:

$$O(R|\vec{x}_k, d_m) = O(R|d_m) \cdot \prod_{\substack{i \\ x_{k_i}=1}} \frac{O(R|x_{k_i}=1, d_m)}{O(R|d_m)} \qquad (BII4)$$

3. The Darmstadt indexing approach

In this chapter, a brief description of the DIA is given. For further details, see [KNOR82] or [FUHR84]. The DIA is a dictionary-based indexing approach for automatic indexing from document titles and abstracts, with a prescribed indexing vocabulary. In the AIR retrieval test ([FUHR83], [FUHR84]) it was demonstrated that the DIA is suited even to large subject fields such as physics.

The indexing task consists of two steps, a description step and a decision step. In the description step, information about the relationship between a descriptor s and the document d to be indexed is collected. This information forms the decision base for the second step in which the probability is estimated, that the assignment of s to d would be correct.

The description step uses an indexing dictionary. The main part of the indexing dictionary consists of weighted relations $r(s,t)$, where the weight approximates the probability $P(C|s,t)$ that the assignment of descriptor s to a document which contains term t would be correct. Terms can be single words, noun phrases or formula identifiers (assigned to formulas by specific algorithms).

The description algorithm starts with the identification of terms in the text. As this task cannot be done perfectly, each term is identified in a certain form of oc-

currence v, where different forms of occurrence correspond to different levels of confidence. If a term t is identified in a document d and a relation $r(s,t)$ is stored in the dictionary, a descriptor indication from t to s is generated. It contains

- the form of occurrence v of t in d,
- the relation $r(s,t)$,
- further information about s, t and d.

The collection of all descriptor indications from a document d leading to the same descriptor s is called the relevance description $y(s,d)$ of s with respect to d.

The decision step uses the relevance description $y = y(s,d)$ to estimate the probability $P(C|y)$ that, if relevance description y is given, the corresponding descriptor assignment would be correct. This estimation is done by the indexing function $a(y)$. Different methods for the development of indexing functions have been investigated for the DIA. In [FUHR85] a probabilistic formula for this purpose is described. Here we will concentrate on the polynomial approach developed by Knorz ([KNOR82,83]) which uses polynomial classifiers adopted in the mean square sense. For this approach, the relevance description y has to to be mapped to a description vector $\vec{y}$. The definition of this mapping has to be done heuristically. Then a coefficient vector $\vec{a}$ is adopted such that $\vec{a} \cdot \vec{y}$ is an estimate of $P(C|\vec{y})$.

The DIA is based on the concept of correctness. For the construction of the indexing dictionary and the adoption of the indexing function, learning samples of documents with correct descriptor assignments must be given. Within the DIA, no assumptions are made about the kind of these samples. For pragmatic reasons, manually indexed documents were used for this purpose in the past. Now experiments with indexing functions adopted on the basis of relevance judgements from retrieval runs have been made for the first time. In this case, the probability $P(C|y(s_i,d_m))$ can be used as an estimation of the probability $P(R|x_{k_i}=1,d_m)$ of the BII model. Retrieval experiments of this kind are described in chapter 7.

4. A general model for retrieval with probabilistic indexing

The retrieval-with-probabilistic-indexing (RPI) model described here is suited to different models of probabilistic indexing. In [ROBE81] a similar retrieval model, the so called two-poisson-independence (TPI) model is described. The main difference between the TPI model and the RPI model is that the RPI model is suited to different probabilistic indexing models, whereas the TPI model is an extension of the two-poisson model for multi-term queries. The TPI model makes more use of the specific assumption of the indexing model, so that for any other indexing model a new retrieval model would have to be developed.

We assume that the event space of the indexing model consists of document-descriptor relationships (ddr), and that a specific ddr is either correct or not. The concept of correctness can be regarded as a pragmatic standard, which differs from one indexing model to another: For most applications of the DIA, manual indexing forms this standard. The same yields for the model proposed in [ROBE84]. In the 2-Poisson model, correctness is equal to the relevance of the document to queries containing only the descriptor considered, and in the BII model the relevance to queries containing this descriptor besides others is regarded. For simplification, we assume in the following that we deal with manual indexing as the correct indexing, so that the probabilistic indexing aims to approximate the manual one.

The RPI model deals with request-document relationships, so its central concept is "relevance". To link the two concepts "relevance" and "correctness" together, the RPI model needs additional relevance infomation about the relationship between a request and the correctness of certain descriptors, that is the relationship between a request and (hypothetical) documents with a certain correct indexing. The event space of the RPI model is - in principle - the same as that of the BII model: all document-request relationships are regarded. In contrast to the BII model, the RPI model is able to distinguish between different requests using the same query formulation. However, as any retrieval system has a restricted knowledge about a request, the notation f_k used in the probabilistic formulas below does not relate to a single request, it stands for a set of requests about which the system has the same knowledge. Equally d_m relates to the system's representation of documents.

An event, a document-request relationship, is regarded with respect to correct indexings. Therefore we use the binary vector $\vec{x} = (x_1, \ldots, x_n)$ where each component corresponds to an element of the set of descriptors S. Here x_i=1 stands for the event that the correct indexing of a document contains descriptor s_i, and x_i=0 in the contrary

case. Using this notation, the probability of correctness estimated by the indexing model can be written as

$$P(x_i{=}1|d_m) = P(C|s_i, d_m)$$

The relevance relationship between a request f_k and the set of documents with a descriptor s_i correctly indexed is described with probabilities of the form $P(R|x_i, f_k)$, that is the probability that a randomly selected document to which s_i was assigned ($x_i{=}1$) resp. not assigned ($x_i{=}0$) correctly is relevant to request f_k.

We denote the set of all possible correct indexings as X, where $|X| = 2^n$. Now we regard the document-request relationship between d_m and f_k with respect to all indexings $\vec{x} \in X$. For the total probability of this event we get

$$P(R|f_k, d_m) = \sum_{\vec{x} \in X} P(R|\vec{x}, f_k) \cdot P(\vec{x}, d_m) \qquad (1)$$

Now we apply Bayes' theorem:

$$P(R|f_k, d_m) = \sum_{\vec{x} \in X} P(R|f_k) \cdot \frac{P(\vec{x}|R, f_k)}{P(\vec{x}|f_k)} \cdot P(\vec{x}|d_m) \qquad (2)$$

The last equation is a general formula for retrieval with probabilistic indexing. Here all dependencies between descriptors can be considered. Before we apply some independence assumptions to simplify this formula, let us have a short look at the different probabilities used here: $P(R|f_k)$ is the probability that a (randomly selected) document would be relevant to request f_k. As this probability is constant for one request, there is no need for its estimation if only a ranking of documents for this request is desired. $P(\vec{x}|R, f_k)$ is the probability that a relevant document (to f_k) has the correct indexing $\vec{x}$. The probability $P(\vec{x}|f_k) = P(\vec{x})$ is independent of the specific request, it stands for the probability that a (randomly selected) document has the correct indexing $\vec{x}$. Finally, $P(\vec{x}|d_m)$ is the probability that the indexing $\vec{x}$ is correct for document d_m.

Now we will make three independence assumptions. As equation (2) is a general formula for retrieval with probabilistic indexing, it is also possible to make assumptions which include certain dependencies between descriptors.

$$P(\vec{x}|d_m) = \prod_{i=1}^{n} p(x_i|d_m) \qquad (A1)$$

$$P(\vec{x}) = \prod_{i=1}^{n} P(x_i) \qquad (A2)$$

$$P(\vec{x}|R, f_k) = \prod_{i=1}^{n} P(x_i|R, f_k) \qquad (A3)$$

With *A1*, we assume that the correctness of a descriptor in a document is independent of the correctness of other descriptors within the same document. This assumption refers to the underlying indexing model, all indexing models mentioned are based on this assumption. *A2* says that the distributions of correct descriptors within the documents are independent of each other. For the relation between relevance and correctness, we assume with *A3* that the correctness of a descriptor in a (randomly selected) document which is relevant to a request f_k is independent of the correctness of other descriptors in this document.

With these assumptions, we get from equation (2):

$$P(R|f_k, d_m) = P(R|f_k) \sum_{\vec{x} \in X} \prod_{i=1}^{n} \frac{P(x_i|R, f_k)}{P(x_i)} \cdot P(x_i|d_m) \qquad (3)$$

This can be transformed into

$$P(R|f_k, d_m) = P(R|f_k) \cdot \prod_{i=1}^{n} \left(\frac{P(x_i{=}1|R, f_k)}{P(x_i{=}1)} \cdot P(x_i{=}1|d_m) + \frac{P(x_i{=}0|R, f_k)}{P(x_i{=}0)} \cdot P(x_i{=}0|d_m) \right) \qquad (4)$$

With the following notations:

$$p_{ik} = P(x_i{=}1|R, f_k)$$
$$q_i = P(x_i{=}1)$$
$$u_{im} = P(x_i{=}1|d_m) = P(C|s_i, d_m)$$

we get

$$P(R|f_k, d_m) = P(R|f_k) \cdot \prod_{i=1}^{n} \left(\frac{p_{ik}}{q_i} u_{im} + \frac{1 - p_{ik}}{1 - q_i} (1 - u_{im}) \right) \qquad (5)$$

In this formula, only descriptors s_i with $p_{ik} \neq q_i$ have an influence on the resulting probability $P(R|f_k, d_m)$. So we can conclude that the query formulation for f_k should exactly contain all these descriptors, which we will denote as set J_k. From the point of view of the indexing model, descriptors with a probability $P(C|s_i, d_m) = 0$ are not assigned to a document. So let I_m be the set of descriptors with $u_{im} > 0$.

Using these two sets of descriptors, we can simplify equation (5) to

$$P(R|f_k, d_m) = P(R|f_k) \cdot \prod_{t_i \in J_k \cap I_m} \left(\frac{p_{ik}}{q_i} u_{im} + \frac{1-p_{ik}}{1-q_i}(1-u_{im}) \right) \prod_{t_i \in J_k \setminus I_m} \frac{1-p_{ik}}{1-q_i} \quad (RPI1)$$

In analogy to Maron & Kuhns' indexing formula, we can make another simplifying assumption: The relevance of a document with respect to a request is not affected by descriptors which occur in the query formulation only but not in the document's indexing:

$$\prod_{t_i \in J_k \setminus I_m} P(x_i=0|R, f_k) \approx \prod_{t_i \in J_k \setminus I_m} P(x_i=0)$$

In this case the second product of formula $(RPI1)$ can be omitted, and we get:

$$P(R|f_k, d_m) = P(R|f_k) \cdot \prod_{t_i \in J_k \cap I_m} \left(\frac{p_{ik}}{q_i} u_{im} + \frac{1-p_{ik}}{1-q_i}(1-u_{im}) \right) (RPI2)$$

To apply the ranking formulas $RPI1$ or $RPI2$, the parameters p_{ik} and q_i must be estimated in addition to the indexing weights u_{im} which come from the indexing model. The values q_i can be derived from the indexing weights in the document collection D:

$$q_i = P(x_i=1) \approx \sum_{d_m \in D} \frac{u_{im}}{|D|}$$

The parameters p_{ik} are request-specific. They can be estimated by using relevance feedback information for a small set of documents D'. Let w_{mk} have the value of the relevance judgement of d_m with respect to f_k with $w_{mk}=1$ if the document is relevant and $w_{mk}=0$ otherwise. Then we can use the following estimation

$$p_{ik} = P(x_i=1|R, f_k) = \frac{\sum_{d_m \in D'} u_{im} w_{mk}}{\sum_{d_m \in D'} w_{mk}}$$

For the experiments described in this paper, global values for all p_{ik}'s and all q_i's were used (adopted from a learning sample of document-request relationships), because there was no appropriate test sample available.

5. Test setting

For the experiments described in the following, the collection from the AIR retrieval test ([FUHR84]) was taken. As this test used boolean search formulations (without NOT operators), retrieval was made in two steps. In the first step, conventional boolean retrieval with queries from the AIR test (which included descriptors only) was performed. For these retrieval runs, a very broad unweighted document indexing was chosen by applying a cutoff-value of 0.01 to the weighted indexing called $A1$ in [FUHR84], which is based on the polynomial approach (in the AIR test a cutoff-value of 0.13 had been used). The sets of output documents selected this way were ranked in the second retrieval step by applying the different ranking formulas described above. The queries for this retrieval step consist of the sets of descriptors from the corresponding boolean queries.

In the first retrieval step, only 244 from the original 309 queries of the AIR test had non-empty answer sets. These 244 queries were divided into three samples named A, B and C. Samples B and C were used for parameter adaption only and sample A was taken for the ranking experiments. Sample A includes 79 queries with 2835 documents, the distribution of answer sizes is shown in table 1. The relevance judgements of the answer documents are values from the relevance scale listed in table 2 (in the original test collection, there were some documents judged as 'non decidable', which have been removed here).

For evaluation, we use the normalised recall measure as defined in [BOLL85] for multistage relevance scales. This measure only considers documents in different ranks and with different relevance judgements. A pair of these

retrieved documents per query	number of queries
1	8
2 – 5	16
6 – 10	11
11 – 20	16
21 – 50	13
51 – 100	10
> 100	5

Table 1: Distribution of answer sizes in the test collection

relevant
conditionally relevant/more relevant
conditionally relevant
conditionally relevant/more irrelevant
irrelevant
digressive

Table 2: Relevance scale

documents is in right order if the document with the higher relevance judgement comes first, otherwise it is in wrong order. Let S^+ be the number of document pairs in right order, S^- the number of those in wrong order, und S^+_{max} the number of documents in right order for an optimum ranking. The normalised recall is defined then as follows:

$$R_{norm} = \frac{1}{2}\left(1 + \frac{S^+ - S^-}{S^+_{max}}\right)$$

A random ordering of documents will have a R_{norm} value of 0.5 in the average. For the cases with S^+_{max}=0 we defined R_{norm}=1. Because of the large scattering of the answer sizes, we use a second average method besides the macro average R^M_{norm}: the micro-macro average R^m_{norm} is a weighted average with respect to the answer sizes. Let n_i be the answer size of retrieval result Δ_i, then the micro-macro average of R_{norm} for a set of t queries is defined as:

$$R^m_{norm}(\Delta_1, \ldots, \Delta_t) = \frac{\sum_{i=1}^{t} n_i \cdot R_{norm}(\Delta_i)}{\sum_{i=1}^{t} n_i}$$

For 12 of our 79 queries (with 43 documents altogether), the value of S^+_{max} equals 0 and so even any ordering of documents will have a value of R_{norm}=1. As a consequence of this, the average R_{norm}-values for random ordering of our test collection are R^m_{norm}=0.505 and R^M_{norm}=0.576.

We use the multivalue relevance scale for evaluation because this yields a finer measure of differences in retrieval quality. In [BOOK81], it is theoretically shown that - under certain conditions - a ranking on the basis of probability of relevance also yields a good ranking according to the degree of relevance. Some experiments which are not described here have shown that the difference between retrieval results remains the same, whether a binary or a multivalue relevance scale is used for evaluation.

6. Estimation of probabilistic parameters

In order to apply the ranking formulas described above three kinds of probabilistic parameters have to be estimated: All formulas are based on weighted indexing, for the BII model, request-independent document weights are required, and the RPI model needs parameters p_{ik} and q_i.

We have developed two kinds of probabilistic indexing for the experiments described in the following. Both indexings are based on the DIA as described in chapter 3 and have been adopted on the basis of the polynomial approach. They only differ in their learning samples with the decisions about correct descriptor assignment:

- Indexing $A1$ was taken from the AIR retrieval test. It was adopted by comparison with manual indexing ([KNOR83b]) using a learning sample of 1000 documents with about 24000 relevance descriptions.
- Indexing $I1$ was adopted on the basis of the retrieval results of the query sample B. According to the BII model, for each descriptors of the query the assignment to a document is assumed to be correct if the document is judged to be relevant or conditionally relevant on the relevance scale used, otherwise the assignment of this descriptor is false. As there were only 2822 documents in this sample, this means that there are only 2822 independent decisions on the basis of which the polynomial approach can be adopted. So this indexing cannot be optimised as far as it is the case with $A1$. The formulas $(BII1)$ and $(BII2)$ additionally require 'absence weights' $P(R|x_i{=}0, d_m)$ for descriptors of a document which do not occur in the query. For $I1$, these indexing weights were estimated in the same way as for the descriptors which were both in the document and in the query.

The request-independent document weights $P(R|d_m)$ also were estimated using the polynomial approach. For this purpose, again sample B was used. The DIA was varied in the way that all descriptor indications from one document formed one 'relevance description', which is judged to be correct only if the document is relevant (or conditionally relevant) to the current request.

For the parameters p_{ik} and q_i, global values were estimated by choosing the best values from some test runs with varied parameters on sample C. This estimation is

not critical, because different pairs (p, q) lead to good results. For the experiments, the values p=0.2 and q=0.15 were selected for both indexings.

7. Experiments

With the different ranking formulas, experiments were made using sample A of the test collection. Table 3 shows the results of different experiments with the BII model.

In experiment no. 1 only the document weights were used for ranking, and it can be seen that these document weights are in fact useful for document ranking. But the results of the experiments no. $2-5$ show that these estimates of $P(R|d_m)$ are not good enough for application of the BII model. For the experiments no. $6-9$, the document weights estimated as described above were replaced by 0.5, the average value of the estimates of $P(R|d_m)$, and this leads to much better results. At the moment these results can not be fully explained. We suppose that the document weight $P(R|d_m)$ and the indexing weights $P(R|x_i, d_m)$ are too much dependent of each other, so that their combination in one formula (based on the assumption that they are independent) does not work. The other remarkable result from experiments no. $2-9$ is that formulas $BII3$ and $BII4$ work significantly better than $BII1$ and $BII2$. The difference between those formulas is that $BII1$ and $BII2$ consider also the descriptors of a document which do not occur in the query, whereas formulas $BII3$ and $BII4$ are restricted to those descriptors which document and query have in common. Other experiments where only a few decriptors with extremly high or low estimate of $P(R|x_i{=}0, d_m)$ were considered also did not show an improvement of the results gained by formulas $BII3$ and $BII4$. So it can be concluded that the 'non-asked' descriptors (ocurring in the document only) do not have any influence on the probability of relevance. Comparing the results of the probability- ($BII1, BII3$) and the odds-formulas ($BII2, BII4$), there is nearly no difference. So one can say that the Maron & Kuhns formula $BII3$ is an approach for the probabilistic indexing which cannot be improved by other assumptions of the BII model.

The R_{norm}-values of the experiments no 10 and 11 from table 3 show that indexing $A1$ is not suited well to the BII model: Although the indexing quality of $A1$ is better than that of $I1$ (see below), the ranking results with the BII model are worse.

The results of the experiments with the RPI model are shown in table 4. Obviously indexing $A1$ fits better to the RPI model than to the BII model. For indexing $I1$, nearly the same results as with the BII model are obtained. This demonstrates that the RPI model can be applied to different kinds of probabilistic indexing. The comparison between the results of formulas RPI1 and RPI2 shows that the consideration of descriptors which occur in the query only (and not in the document) brings slight improvements of the ranking results.

In experiments no 16 and 17, the cosine measure was used for document ranking. There is no great difference to the results obtained from the probabilistic ranking procedures.

no	formula	doc.weight	indexing	R^M_{norm}	R^m_{norm}
0	random ord.			0.576	0.505
1		$P(R\|d_m)$		0.637	0.541
2	$BII1$	$P(R\|d_m)$	$I1$	0.569	0.495
3	$BII2$	$P(R\|d_m)$	$I1$	0.581	0.493
4	$BII3$	$P(R\|d_m)$	$I1$	0.662	0.649
5	$BII4$	$P(R\|d_m)$	$I1$	0.671	0.657
6	$BII1$	0.5	$I1$	0.657	0.552
7	$BII2$	0.5	$I1$	0.657	0.545
8	$BII3$	0.5	$I1$	0.733	0.700
9	$BII4$	0.5	$I1$	0.732	0.702
10	$BII3$	0.5	$A1$	0.732	0.685
11	$BII4$	0.5	$A1$	0.716	0.676

Table 3: Results of experiments with the BII model

no	formula	indexing	R^M_{norm}	R^m_{norm}
12	*RPI*1	*I*1	0.727	0.699
13	*RPI*2	*I*1	0.729	0.701
16	*cosine*	*I*1	0.709	0.701
14	*RPI*1	*A*1	0.770	0.733
15	*RPI*2	*A*1	0.762	0.721
17	*cosine*	*A*1	0.769	0.731

Table 4: Results of experiments with the RPI model and the cosine measure

Finally, comparing the different results for *I*1 and *A*1 in table 4, it can be seen that indexing *A*1 provides better retrieval results than *I*1. We assume that the main reason for this difference is that the learning sample used for the adoption of I1 is too small.

5. Conclusions

The results described in the previous section show that probabilistic indexing can successfully be used for ranking procedures. The DIA forms the basis for the application of different probabilistic indexing models: With the DIA, it is possible to estimate probabilistic indexing weights required for the Maron & Kuhns model, and it has been shown that this model yields good ranking results. Here it should be emphasised that the indexing dictionary which has been used for the development of the *I*1 indexing was the same as that used for *A*1, which has been derived from manual indexing. At the moment, there is no possibility to build up a dictionary on the basis of retrieval results, because there is not enough data available. But all the ranking results show that the adaption of automatic indexing on the basis of manual indexing, the proceeding that has been chosen for the experiments with the DIA so far, is a practicable and successful method for the development of automatic indexing systems.

For the RPI model, which has been proposed in this paper, it has been shown that this model is suited to different kinds of probabilistic indexing. It is more flexible then the BII model, because it works with two concepts: 'correctness' as a basis of the underlying indexing model, and 'relevance' for the retrieval parameters. In this paper, no experiments have been made with request-specific parameters p_{ik} and q_i as a kind of search term weights. The combination of probabilistic indexing and search term weighting is also the main objective of the unified model described a few years ago ([ROBE82]). So it would be interesting to compare both models in experiments.

The comparison of the probabilistic ranking formulas with the cosine measure has shown no difference in this paper. Of course the application of the cosine measure is easier, but the probabilistic models have the advantage of being more transparent, because the underlying assumptions are made explicit. Obviously these assumptions do not fit the reality, but there is the possibility to replace them by appropriate assumptions, in order to achieve better retrieval results.

References

[BOOK74] Bookstein, A., Swanson, D.R., "Probabilistic models for automatic indexing", Journal of the American Society for Information Science 25, pp. 312-318.

[BOOK75] Bookstein, A., Swanson, D.R., "A Decision Theoretic Foundation for Indexing", Journal of the American Society for Information Science 26, pp. 45-50.

[BOOK83] Bookstein, A., "Outline of a General Probabilistic Retrieval Model", Journal of Documentation 39(2), pp. 63-72.

[FUHR84] Fuhr, N., Knorz, G., "Retrieval test evaluation of a rule based automatic indexing (AIR/PHYS)", Rijsbergen, C.J. van (Ed.): Research and Development in Information Retrieval, pp. 391-408. Cambridge University Press, Cambridge.

[FUHR85] Fuhr, N., "A probabilistic model of dictionary based automatic indexing", Proceedings of the riao 85, Recherche d'Informations Assistée par Ordinateur, pp. 207-216.

[HART75] Harter, S.D., "Probabilistic Approach to Automatic Keyword Indexing, Part I: On the Distribution of Speciality Words in a Technical Literature. Part II: An Algorithm for Probabilistic Indexing", Journal of the American Society for Information Science 26, pp. 197-206, 280-289.

[KNOR83a] Knorz, G., "Automatisches Indexieren als Erkennen abstrakter Objekte", Sprache und Information, Band 8. Niemeyer, Tübingen.

[KNOR83b] Knors, G., "Development of automatic indexing for the AIR retrieval test", Experiments by means of ALIBABA. Internal Report DVII 83-3, TH Darmstadt, Fachbereich Informatik, Fachgebiet Datenverwaltungssysteme 2.

[MARO60] Maron, M.E., Kuhns, J.L., "On Relevance, Probabilistic Indexing, and Information Retrieval. Journal of the Association for Computing Machinery 7(3), pp. 216-44.

[MARO82] Maron, M.E., "Probabilistic Approaches to the Document Retrieval Problem", Salton, G., Schneider, H.-J.(Ed.): Research and Development in Information Retrieval. Springer, Berlin, Heidelberg, New York, pp. 98-107.

[ROBE81] Robertson, S.E., van Rijsbergen, C.J., Porter, M.F., "Probabilistic models of indexing and searching", (Ed.: Oddy, R.N; Robertson, S.E.; van Rijsbergen, C.J.; Williams, P.W.) Information Retrieval Research, Butterworth London, pp. 35-56.

[ROBE82] Robertson, S.E., Maron, M.E., Cooper, W.S., "Probability of relevance: A unification of two competing models for document retrieval. Information Technology: Research and Development 1, pp. 1-21.

[ROBE84] Robertson, S.E., Harding, P., "Probabilistic automatic indexing by learning from human indexers", Journal of Documentation 40(4), pp. 264-270.

Probabilistic Models for Document Retrieval: A Comparison of Performance on Experimental and Synthetic Databases

Robert Losee
U. of North Carolina, Chapel Hill, NC. 27514

Abraham Bookstein
U. of Chicago, Chicago, IL. 60637

Clement Yu
U. of Illinois, Chicago, IL. 60680

Abstract

Probabilistic document retrieval systems consistent with the two Poisson independence model outperforms the binary independence model if the terms are distributed as described by the model's assumptions. The Two Poisson Effectiveness Hypothesis suggests that retrieval models based upon the two Poisson model will outperform binary independent models when used on a "real-world" database, where independence and two Poisson term occurrence distributions fail to hold, because the added information obtained from incorporating term frequency information will more than compensate for the non-Poisson distributions of terms. Searches of the MED1033 database suggest that if terms are not independent and frequencies of term occurrence are not distributed in a two Poisson manner, the binary independence sequential retrieval model outperforms the two Poisson independence retrieval model.

1. Introduction

Documents are normally written "about" a particular subject or subjects and are said to be relevant to the subject, while documents which "satisfy" a subject query are said to be relevant to the query. The human mind can rapidly determine the subject of a text fragment; it is more difficult to design an algorithm that performs the same task. Such algorithms often take advantage of differences in term distributions in documents relevant to a subject and in documents not relevant to the subject; probabilistic models may formally describe these distribution. Two models have been used with some success: the binary independence and two Poisson independence models.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

2. Binary Independence Assumptions

Features which may take only the values 0 or 1 are said to be binary features. They may indicate that (1) a term is present; (2) a document is relevant to a subject; (3) a document cites a particular document; (4) a document is cited by a particular document; or (5) a document has been retrieved or circulated by a particular patron.

We may simplify our term distribution model if we assume term independence, that is, if we believe that the occurrence or non-occurrence of a given feature in a document provides no information concerning the probability of occurrence of any other feature in the same document. If we combine our binary and independence assumptions, the following distribution describes feature occurrences among binary terms of a given relevance class:

$$Pr(d|p) = p^{d}(1 - p)^{1 - d},$$

where p represents the probability of feature d occurring. Because we do not have perfect knowledge concerning the value of the parameter p for each relevance class, we may describe a distribution, g(p), characterizing p. The choice of a conjugate distribution for our g(p) will simplify future calculations. The conjugate distribution for the binary independence model is the Beta distribution,

$$p^{S-1}(1-p)^{R-S-1},$$

which we will use to represent our knowledge of p. The expected value of the Beta distribution is $E(p|S,R) = S/R$; while the variance of the Beta distribution is $V(p|S,R) = (S(R-S))/(R^2(R+1))$. As our knowledge of p increases, the variance of

g(p) decreases; our distribution develops a sharper peak, representing a higher degree of confidence concerning p.

3. Two Poisson Independence Assumption

Natural language text consists of features which may occur any number of times. The Poisson term frequency distribution model has several desirable characteristics that suggest its use in describing the frequency of term occurrences, including its generation of non-negative integers, its memoryless property, and its mathematical tractability; it may thus be considered as a suitable candidate to model term distributions [HART74, BOOK75]. A Poisson term generation process may be seen, informally, as the random production of term occurrences as text is generated where there is a small probability of any individual term occurring at any time. More formally, the Poisson assumption states that the conditional probability of d occurrences of a term in a document, given an expected frequency u, is

$$Pr(d|u) = \exp(-u)u^{d}/d!,$$

if we again make the assumption of term independence.

The conjugate distribution for the Poisson distribution, the gamma-1, represents our knowledge of u. Omitting terms that ultimately cancel, g(u) is

$$e^{-u(R-1)}u^{S},$$

S and R are parameters interpreted below. The mean of the gamma-1 distribution is S/R, while the variance is S/R^{2}.

4. Sequential Learning Model

The sequential learning model has the capability to "learn" to recognize the characteristics of relevant and non-relevant documents from relevance feedback consisting of user document relevance judgments [BOOK83]. As was mentioned above, certain probability distributions are functions of the document profiles (d), the set of features describing a document. These distributions may be characterized by a vector parameter P, which is not known exactly and is itself also described by a probability distribution, g(P). The sequential learning model suggests that, after n stages of feedback, documents be ranked by a criterion function

$$\prod_{i=1}^{q} \frac{f_{Ri}(d_i;n)}{f_{Ri}(0_i;n)} \cdot \frac{f_{Ni}(d_i;n)}{f_{Ni}(0_i;n)},$$

where $f_{Ri}(d_i;n) = \int g_{Ri}(P;n)\, f_{Ri}(d_i|P)\, dP$ for relevant documents; g(P;n) denotes the distribution g(P) after n feedback stages, and $f_{Ni}()$ is a similar function for non-relevant documents. The value returned by the criterion function is the Retrieval Status Value (RSV).

When a system based upon the sequential learning model begins operating, all the documents are evaluated using the criterion function and those documents with the highest values are retrieved. The parameters for our prior distribution, g(P) (for example, S and R above), represent our initial knowledge about P; optimal initial parameters for this distribution in the class of relevant and non-relevant documents are unknown but are being studied. After the retrieval of a set of documents, the user provides relevance feedback from which the system "learns," thereby modifying the function g(P) to g(P;n). The system then evaluates the remaining documents with a criterion function incorporating this learning; the user retrieves another set of documents, and the process repeats itself.

5. Binary Independence Retrieval Models

We may incorporate the binary independence assumptions with the sequential learning model. If we do so, $f_{R}(d|n)$ becomes

$$\int p^{d}(1-p)^{1-d}p^{(S+s)+d-1}(1-p)^{(R+r)-(S+s)-d}dp$$

ignoring portions of the formula eventually canceling. s is the number of all the previous binary d_i's occurring in the retrieved relevant documents, r is the number of previously retrieved relevant documents; d indicates whether the feature whose probability distribution is being described occurs.

The parameters R and S have an intuitively satisfying interpretation. R represents the degree of confidence, before any retrievals have been made, in an expected frequency of occurrence of a term for relevant documents. A small number represents little confidence and a large number represents a great deal of confidence. After n feedback stages, for notational simplicity, we set R' = (R+r); R' the confidence parameter after feedback, is thus measured by the number of relevant document actu-

ally seen plus the initial parameter R, which effectively acts as the number of documents imagined to have been seen before the search began. S, and S' = (S+s), represent the frequency of occurrence per document expected in the set of relevant documents multiplied by the confidence factor R'; we are acting as if we saw R' relevant documents, of which r are actual and R fictitious. The corresponding parameters for non-relevant documents are preceded by the letters "N" or "n." The integral evaluates to

$$\frac{((S+s)+d-1)!\ ((R+r)-(S+s)-d)!}{(R+r)!}.$$

The Retrieval Status Value based on the above formula is

$$\prod \frac{(S'+d-1)!\ (R'-S'-d)!}{(S'-1)!\ (R'-S')!} \Big/ \frac{(NS'+d-1)!\ (NR'-NS'-d)!}{(NS'-1)!\ (NR'-NS')!}.$$

As S' and R'-S' become large, the expression approaches

$$\sum d \log \frac{S'/R'}{1-(S'/R')} \Big/ \frac{NS'/NR'}{1-(NS'/NR')}.$$

6. Two Poisson Independence Retrieval Model

Similarly, we may incorporate the two Poisson independence assumptions into the sequential learning model. If we do so, after integration, and as above,

$$f_R(d|n) = (S+s+d)!\ /\ d!(R+r)^{S+s+d+1},$$

where S and R are the initial parameters for the gamma-1 distribution, s is the sum of all the previous d_i's occurring in the retrieved relevant documents, r is the number of previously retrieved relevant documents; d is the number of occurrences of the feature, whose probability distribution is being described. The corresponding parameters for non-relevant documents are preceded by the letters "N" or "n" . Subscripts indicating which term is used are dropped for notational simplicity. The full criterion function becomes

$$\sum \log \frac{(S+s+d)!}{(S+s)!} \frac{(NS+ns)!}{(NS+ns+d)!} + \sum d \log \frac{(NR+nr)}{(R+r)}$$

This expression can be simplified if S+s (and the corresponding expression for non-relevant documents) is large enough that log x does not vary much from S+s to S+s+d; if so, the criterion function can now be approximated by

$$\sum d \log (u/v),$$

where u = (S+s)/(R+r) and v = (NS+ns)/(NR+nr). u and v are the means of gamma-1 distributions estimating the expected frequency of term occurrences in relevant and non-relevant documents.

7. Parameter Estimation

The sequential learning model suggests that documents be retrieved in the order of their descending RSVs, which are a function of four values: S', R', NS', and NR'. S, R, NS, and NR are the values for the model parameters and s, r, ns, and nr are values representing knowledge obtained from retrieved documents; specifically: s (ns) is the number of term occurrences in relevant (non-relevant) documents retrieved, and r (nr) is the number of relevant (non-relevant) documents retrieved. The expected frequency of term or feature d in a relevant document, under both binary independence and two Poisson independence assumptions, is

$$S'/R' = E(d|\text{relevance}),$$

while the expected frequency of a term's occurrence in a non-relevant document, E(d|non-relevance), is NS'/NR'.

Learning after retrieval that a document is relevant results in r, and thus R', incrementing by 1 and s, and thus S', increasing by the term frequency d. Thus, if S=1, R=2, S/R = 1/2, and if the first relevant document contains 1 occurrence of the term in question, S'/R' will become 2/3. If S = 2 and R = 4 (so S/R = 2/4) and the same document was retrieved, S'/R' would become 3/5. We can informally see that the value of the initial parameters, the initial knowledge, affects the final knowledge.

8. Croft-Harper Estimates

Croft and Harper have suggested that the probability that a binary term occurs in non-

relevant documents may be estimated from the frequency of term occurrences in the entire database, under the assumption that most of the documents in the database are non-relevant [CROF79]. Given the value of parameter NR, E(d|non-relevance) * NR is denoted below as CH and represents a form of the Croft-Harper estimate for NS.

The expected value of the number of occurrences of a term for the non-relevant documents under the two Poisson independence assumption may be estimated in a similar manner. The averages over documents for each term in the database may be used as the expected frequencies for non-relevant documents.

A test database such as MED1033 may be presumed to have a ratio of relevant to non-relevant documents which is much higher than would be found in a commercial database with potentially millions of non-relevant documents for every relevant document. This suggests that the Croft-Harper estimation procedure will perform better with a commercial database than with our test database, with its higher proportion of relevant documents "tainting" the estimate of E(d|non-relevant). Thus, any negative results obtained regarding the Croft-Harper estimates should be moderated by considering the effect of our experimental database on the results.

9. Experimental Database

The Medlars database with 1033 documents (MED1033) has been used for this study. It has been assembled and placed in its present form by Gerald Salton and his colleagues at Cornell University's Computer Science Department for use with the SMART document retrieval system. On the basis of query, title, and abstracts, MED1033 relevance assessments were "made by veterinary students ... at Cornell" who were presented with document abstracts and titles which were exhaustively evaluated [VOOR84].

The MED1033 documents were selected by a process unknown to the author from the National Library of Medicine's databases. Natural language forms of document abstracts and titles were indexed by the SMART system at Cornell University. Common terms from a "stop list" were removed from titles and abstracts and the remaining terms were "stemmed" using standard SMART stemming procedures. Eight thousand, seven hundred and fifty stemmed terms are used to index the collection, with an average of 55.8 terms per document [FOX83]. The information available for each document includes the applicable term stems and the number of times each occurs in the document. Such frequency information is available for use by systems based on non-binary models of document profiles such as the two Poisson independence model. For systems based on the binary independence model, terms with frequencies of 1 or more are assigned a binary 1 and terms with frequency 0 are assigned a binary 0.

Our research compares results using the MED1033 database with that using a synthetic database, S, constructed, as described below, to fulfill our theoretical assumptions.

10. Precision and Recall Measurements

Document retrieval systems retrieve documents of differing values to the user. It is the function of system performance measurements to provide numbers to be used in comparing the effectiveness of different retrieval methods or parameters, that is, "the ability of the system to retrieve relevant documents while at the same time holding back non-relevant ones" [VANR79]. The most frequently used measurements are based on recall and precision. Recall may be defined as the probability that a document has been retrieved given that it is relevant or, in a sequential search, as the percent of relevant documents already retrieved. Precision may be defined as the probability that a document is relevant given that it is retrieved or, in a sequential search, as the percentage of documents retrieved that are relevant. Precision and recall values are often combined into a precision-recall curve, a smoothed curve connecting the different recall and precision points for a particular search summarizing a group of searches. The average recall-precision curve usually shows decreasing precision as recall increases, that is, the further a search progresses, the lower will be the likelihood that a relevant document will be retrieved next.

Experimental results based on the recall-precision curve will be given in Appendix 1. Results for each retrieval decile from the ten percent retrieval level to the ninety percent level will be given as follows:

1BS. 87 75 65 58 50 46 39 30 21 51.3

where "1BS." represents the experiment number, whether the search is binary (B) or two Poisson (P), and whether the MED1033 (M) or a synthetic (S) database is used. The number at the right is the average of the precision at 25, 50, and 75 percent levels and is computed before any rounding. The average provides an easy to interpret, single number measure. An explanation follows the average in some cases, providing identifying information about the experiment. The explanation will contain, in braces, the values of

parameters defining the particular run in the form (S,R,NS,NR).

11. Accuracy of Statistical Assumptions

Feature occurrences in documents and queries are not statistically independent as is assumed by the binary independence and two Poisson independence models. A retrieval system based on a statistical model making inaccurate assumptions, for example, the binary independence and two Poisson independence models, may be expected to perform in a sub-optimal fashion. The degree to which independence models fail may be determined by comparing results of searches of actual databases with results of searches of databases synthesized so term independence assumptions hold.

Given the number of term occurrences for each term and for each query in the MED1033 database, we will produce four databases with independent term frequencies by using random number generators to produce term occurrences. The random number generator will produce term occurrences distributed consistent with the two Poisson independence hypothesis, allowing an examination of not only the degree to which independence assumptions affect performance but the degree to which the two Poisson independence assumptions hold.

Data is stored in the experimental system in tuples with a primary key containing query number, term number, and document number [ULLM82]. The expected number of term occurrences in both relevant and non-relevant documents will differ in our random databases for a given term from query to query. For example, the term "eye", occurring in both queries 1 and 2, may have an average of 3 occurrences in documents relevant to query 1 and an average of only 1.5 occurrences in documents relevant to query 2. Our generated database must be able to have terms distributed with different model parameters for different queries. When term frequencies are randomly generated for the term "eye" in our example above, the expected frequency for "eye" in relevant documents for query 1 will be 3, while for query 2 it will be 1.5.

A Poisson random number generator takes as its input an expected value, u, and produces an integer greater than or equal to 0. The m parameter is a function of the query number, the term number, and the relevance class in MED1033. The mean may be obtained for relevant and non-relevant documents for a particular term, and these means are used in the generation of terms in both relevant and non-relevant documents for the query in question.

A linear congruential generator may be used to generate real numbers in the range from 0 to 1 [BRAT83]. This random number can then be used by a Poisson random number generator using the following algorithm:

```
x = -1
m = exp(u)
repeat
    generate uniform random U
    x = x + 1
    m = m * U
until m < 1
output x
```

Binary profiles will be obtained by assuming that all term frequencies of 0 represent a binary 0, while all other frequencies represent a binary 1.

12. Binary Independence versus Two Poisson Independence Models

The binary independence model makes 2 assumptions: binary term frequencies and term independence in relevant and non-relevant documents; the term independence assumption is obviously false, but may be a useful approximation of the true level of dependence. The two Poisson Independence model, on the other hand, assumes both term independence and Poisson distributed terms in both relevance classes; both assumptions are likely to be inaccurate. We may now state a Two Poisson Effectiveness Hypothesis:

> Even though terms are not Poisson distributed and independent, the term frequencies are distributed in a manner close enough to the Poisson distribution that the degradation due to the failure of the distribution assumption to be met is compensated for by the increase in information provided by non-binary term frequencies. Retrieval models using the two Poisson independence assumption will yield superior results to those based on binary independence assumptions.

If this hypothesis is correct, search results based on the two Poisson independence model should be superior to results from searches based on the binary independence model. A comparison of binary independence and two Poisson independence search results with ad hoc estimates of S, R, and NR parameters and CH estimates of NS are shown below (Appendix 1) in 1BM and 1PM, while searches with S/R fixed to the average S/R ratio for the entire database are given in 2BM and 2PM, using the highest average precision available for the respective models. These results suggest that searches based on the binary independence model consistently perform as well as or better than searches based on the two Poisson independence model.

This ordering of model performance is consistent with results reported by Raghaven, Shi, and Yu [RAGH83 - Table 5]. These results strongly argue against accepting the Two Poisson Effectiveness Hypothesis, which claims the superiority of the two Poisson model.

When trying to understand these results, we might note that the binary independence model is better during the earlier phases of the search, with performance sometimes dropping to below that of the two Poisson independence model. A possible explanation for this drop is that higher frequency terms, usually retrieved first with a two Poisson independence model, deviate more from a Poisson distribution than do the lower frequency terms. It should be noted that the use of the binary independence model with a tie breaking heuristic, used in 2BM, consistently outperforms the two Poisson independence model at all recall levels [LOSE86].

13. Results with Random Databases

Four random databases were generated, as per the techniques described above, with seeds of 131, 1985, 12345, and 332211. Each document has a Poisson distributed value for each term vis à vis each request. If a term occurs more than once, it is given a value of one for the binary independence model. Binary independence results for a random database with the seed of 1985 are given in 3BS and 4BS, while results for a two Poisson independence search are given in 3PS and 4PS. The average precision of two Poisson independence searches is 15% to 25% above that of the binary independence searches of the synthetic database, supporting the Two Poisson Effectiveness Hypothesis claim that a model accurately describing non-binary term frequencies will outperform one accurately describing binary term frequencies.

Performance under the binary independence model increases dramatically when the distribution of terms is truly independent, as it is in the random database. Using retrieval algorithms consistent with the two Poisson assumption results in performance increases over the binary independence models when the synthesized random database is used. The two Poisson, independently distributed term model provides more information about the probability of relevance of documents than the binary independently distributed term model; thus, if the model more accurately describes the data, retrieval models based on the two Poisson independence data should out-perform binary independence searches using binary independence data. We would thus expect the two Poisson independence searches to have higher average precisions than the binary independence searches when using a random database; our experiments show this is the case.

The excellent results obtained with the two Poisson independence model and the database meeting the assumptions of the model may be achieved with a range of other models, such as a Negative Binomial term distribution model. A model which accurately describes the distribution of terms and also accurately describes the dependencies between terms may be expected to perform near the level obtained in our experiment. Experimental evaluation of the MED1033 term frequency statistics for relevant and non-relevant documents shows that for 73.7% of the terms occurring in relevant documents and for 93.2% of the terms occurring in non-relevant documents, the variance was greater than the mean. This is contrary to what we would expect if terms were Poisson distributed in both classes of documents; a Poisson distribution has an equivalent mean and variance. The Negative Binomial distribution allows for the variance to exceed the mean and will be the object of future study.

14. Measuring the Effect of Assumption Failure

Experiments 3 and 4 were obtained with searches of one synthetic database, with the three other databases generated using different seeds yielding similar orderings (Figure 1). This supplies evidence for the proposition that the two Poisson independence model will outperform the binary independence model, given two Poisson distributed data.

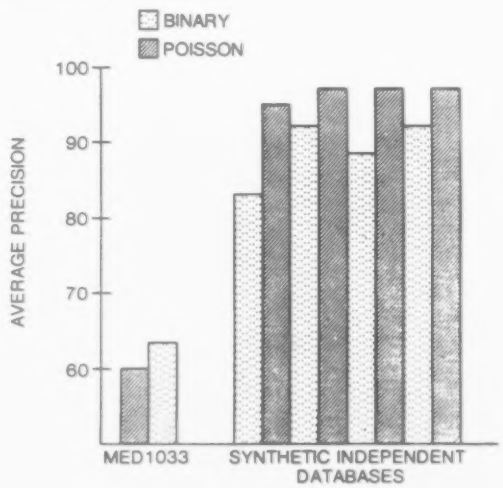


FIGURE 1

Comparative performance of BI and TPI models on realistic (MED1033) database and 4 synthesized databases consistent with TPI assumptions.

Figure 1 also suggests, using the notation of our precision recall measurements, that searches are strongly ordered:

PS > BS > BM > PM

We denote the average precision of a search within bars, for example, |BS|. The decrease in binary performance due to the failure of the independence assumption to hold is approximately |BS| - |BM| = Δ I. The performance decrease due to the failure of the two Poisson assumption may now be approximated by |TS| - |TM| = Δ I = Δ P. Figure 1 suggests that Δ I is about 25 percentage points, while Δ P is about 8 percentage points.

15. Summary

Document retrieval systems consistent with the two Poisson independence model outperforms the Binary independence model if the terms are distributed as described by the models. The MED1033 database, however, has terms distributed in such a manner that the model consistent with the two Poisson independence assumptions fails to perform as well as the model consistent with the binary independence assumptions. This suggests that the Two Poisson Effectiveness Hypothesis, which claims that increases in system performance due to the incorporation of term frequency data should more than offset the decrease due to the non-Poisson distribution of terms, is false.

References

Bookstein, A. and Swanson, D. "A Decision Theoretic Foundation for Indexing." Journal of the American Society for Information Science. XXVI (January 1975): 45-50.

Bookstein, A. "Information Retrieval: A Sequential Learning Process." Journal of the American Society for Information Science. XXXIV (September 1983): 331-342.

Bratley, P., Fox, B., and Schrage, L. A Guide to Simulation. (New York: Springer-Verlag, 1983).

Croft, W. and Harper, D. "Using Probabilistic Models of Document Retrieval without Relevance Information." Journal of Documentation. XXXV (December 1979): 285-295.

Fox, E. Characterization of Two New Experimental Collections in Computer and Information Science Containing Textual and Bibliographic Concepts. Technical Report 83-561, Cornell University Department of Computer Science. Ithaca, New York: September, 1983.

Harter, S. "A Probabilistic Approach to Keyword Indexing." Ph.D. dissertation, University of Chicago, 1974.

Losee, R. "The Performance of Probabilistic Models of Document Retrieval Systems." Ph.D. dissertation, University of Chicago, 1986.

Raghaven, V., Shi, H. and Yu, C. "Evaluation of the 2 Poisson Model as a Basis for using Term Frequency Data in Searching." Proceedings of the Sixth Annual International ACM SIGIR Conference on Research and Development in Information and Retrieval. (New York: Association for Computing Machinery, 1983).

Ullman, J. Principles of Database Systems, second edition. (Rockville, Maryland: Computer Science Press, 1982).

Van Rijsbergen, C. Information Retrieval, second edition. (London: Butterworths, 1979).

Voorhees, E. Computer Science Department, Cornell University, Ithaca, New York. Letter of 18 June, 1984 and personal communication of 19 June, 1985.

APPENDIX 1

1BM.	78.8	75.7	74.2	69.2	60.8	52.1	42.9	34.2	27.0	58.1	(1,2,CH,300)
1PM.	76.6	73.5	70.3	65.3	58.2	51.4	43.3	37.7	32.2	56.9	(1,2,CH,300)
2BM.	80.2	74.9	74.9	71.3	65.6	59.1	54.3	46.0	38.4	63.6	(.25,1,CH,100)
2PM.	77.4	73.1	70.4	64.5	57.5	50.7	42.8	38.7	31.9	56.7	(.66,4,CH,100)
3BS.	80.4	78.2	79.4	79.0	74.8	73.6	65.6	58.5	52.2	71.9	(1,2,CH,2)
3PS.	100.0	99.5	97.9	97.9	95.5	93.7	90.8	83.6	70.5	93.8	(1,2,CH,256)
4BS.	82.7	79.7	77.0	75.6	72.8	69.4	63.6	58.8	49.0	70.8	(5,8,CH,8)
4PS.	100.0	99.3	97.5	97.2	95.8	93.2	90.6	83.5	67.7	93.7	(5,8,CH,8)

NON-BINARY INDEPENDENCE MODEL

by

C. T. Yu and T. C. Lee

Department of Electrical Engineering & Computer Science
University of Illinois at Chicago
Chicago, Illinois, 60680 USA

INTRODUCTION

In the binary independence model [YuSa], the document vectors are binary vectors and a weight, known as the term precision weight is assigned to each query term. It is shown in [YuSa] that its retrieval performance will be better than that of assigning equal weights to all query terms. By taking the logarithm of the term precision weight, it is shown in [RoSp] that retrieval is optimal under the binary independence model.

When a term occurs more than once in a document, there is a loss of information if its presence but not its actual number of occurrence is recorded (as in the binary independence model). Thus, it is beneficial to make use of the additional frequency information to obtain better retrieval performance than the binary independence model. Previous attempts to make use of the frequency information gave little or no improvement [RaSY,Lose], when the 2-Poisson model [BoSw,Hart] is applied. However, an ad hoc method in [RaSY] using a technique in [Crof] yields some improvement.

In this paper, the distributions of the terms in the set of relevant documents and their distributions in the set of irrelevant documents are given. Based on the given distributions, weights are assigned to terms in the documents (unlike the situation in which weights are assigned to query terms in the binary independence model). It is shown that the assignment yields optimal retrieval. Experimental results using two actual collections show that there is a significant improvement in retrieval performance over the binary independence model.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

2. NON-BINARY INDEPENDENCE MODEL

The binary independence model makes the following two assumptions

(1) The occurrence of a term in a document is either 0 or 1. 1 indicates its presence; 0 its absence.

(2) The distributions of terms in the set of relevant documents of a given query are independent; their distributions in the set of irrelevant documents of the given query are also independent.

The non-binary independence model retains assumption(2), but allows a term to occur zero or more times in a document. If a term occurs i times, i > 1, in a document, then it is likely that its significance is higher than if it occurs only once in the document. By making use of the additional information, we should obtain better performance than the binary independence model.

Let the given query Q be (1,1,...,1) i.e. it contains the first n terms and the terms that are not present in the query are ignored. Let the frequency occurrence of the set of n terms in a document be represented by a vector of non-negative integers X = (x1,...,xn), where xi represents the number of occurrences of term i in the document. Let the document be represented by X' = (x1',...,xn'), where xi' represents the weight of term i in the document. For binary independence model, X and X' are the same, but they are different in non-binary independence model. Let R and I denote relevance and irrelevance respectively and the f be a similarity function. By [Robe], the optimal way to retrieve documents is to retrieve them in descending order of relevance i.e. for any two documents Di and Dj,

$$P(R \mid Di) > P(R \mid Dj) \iff f(Q,Di') > f(Q,Dj') \quad (1)$$

where P(|) denotes conditional probability and Q is the given query. Thus, a document with a higher degree of relevance should be retrieved ahead of another document with a lower degree of relevance. By Bayes's theorem,

$$P(R \mid Di) = [P(Di \mid R)\,P(R)]/[P(Di \mid R)\;P(R) + P(Di \mid I)\,P(I)] \quad (2)$$

By substituting (2) into L.H.S. of (1), we obtain

$$P(R \mid Di) > P(R \mid Dj) \text{ iff } P(Di \mid R)/P(Di \mid I) > P(Dj \mid R)/P(Dj \mid I)$$

Let $Di = (di1,\ldots,dik,\ldots,din)$ and $Dj = (dj1,\ldots,djk,\ldots,djn)$. By making use of the term independence assumption i.e. assumption(2), the above inequality reduces to

$$\prod_{k=1}^{n} [P(dik \mid R)/P(dik \mid I)] > \prod_{k=1}^{n} [P(djk \mid R)/P(djk \mid I)]$$

By taking log to both sides, the inequality becomes

$$\sum_{k=1}^{n} \log [P(dik \mid R)/P(dik \mid I)] > \sum_{k=1}^{n} \log [P(djk \mid R)/P(djk \mid I)]$$

Let the similarity function f be the simple dot product, i.e.

$$f((x1,\ldots,xn),(y1,\ldots,yn)) = \sum_{k=1}^{n} xi.yi$$

The weight of the kth term in document Di, dik', is given the assignment

$$\log [P(dik \mid R)/P(dik \mid I)] \quad (3)$$

$$f(Di',Q) = \sum_{k=1}^{n} \log [P(dik \mid R)/P(dik \mid I)].$$

Thus, the inequality (1) is satisfied by the term weight assignment (3)

And therefore optimal retrieval under the non-binary independence model is guaranteed.

Note: The result obtained here is somewhat different from that obtained in the binary independence model. In the binary independence model, query terms are assigned weights, but document vectors are unchanged. In the non-binary independence model, the query vector is a binary vector (with zeros after the nth positions) but the document vector $D = (d1,\ldots, dk,\ldots,dn)$ is modified to become $D' = (d1',\ldots,dk',\ldots,dn')$ with dk' reflecting the significance of term k in document D.

3. EFFICIENCY CONSIDERATION:

We first sketch how similarity values are computed for the binary independence model using inverted files. Then, the approach is generalized to handle the non-binary independence case.

In the binary independence model, an inverted file exists for each term. For each term, the file contains the set of numbers associated with the documents having the term (see figure 1). Thus, documents which do not contain the term are not included in the file.

Given a query $Q = (q1,\ldots,qn)$, where all zero components are eliminated and the non-zero components are moved to the first n positions. f(Q,Di) is set to be zero for each Di initially.

Figure: 1

	t1	t2	t3
D1 =	(1,	0,	0)
D2 =	(1,	1,	0)
D3 =	(0,	0,	1)
D4 =	(0,	1,	0)

Inverted files for

t1 : {1, 2}, t2 : {2, 4}, t3 : {3}

Since $f(Q,Di) = \sum_{k=1}^{n} qj\,dj$, $Di = (d1,\ldots,dn)$, where dj is in {0,1}. it can be rewritten as $\sum qj$, where the summation is over all terms tj in Di. The inverted files of the n terms of Q are scanned. For each term tj, f(Q,Di) is increased by qj if the corresponding file contains Di. Thus, the procedure correctly computes the similarity for f(Q,Di) for each Di. The main point is that most entries in a document vector are zeros. Since they are not stored in the inverted files, they will not be scanned and therefore substantial saving in time and space will be obtained.

In the non-binary independence model, each term tk in document Di is given the weight dik' = log [P(dik | R)/P(dik | I)]. Even if dik = 0, dik' is usually not zero. As a result, the document contain mostly non-zero entries and therefore the corresponding inverted files are large. In order to save space and time for the computation of the similarity values, we redefine the weight of term tk in document Di, dik'', to be

$$\log[P(dik \mid R)/P(dik \mid I)] - \log[P(0 \mid R)/P(0 \mid I)] \quad (4)$$

i.e. dik'' is the relative weight from the case where dik = 0. Clearly when dik is 0, dik'' = 0 and therefore need not be stored in the inverted files.

We now show that, with the new weight assignment, optimal retrieval is preserved.

Proposition: f(Q,Di') > f(Q,Dj') iff f(Q,Di'') > f(Q,Dj''), where the weights in Di' and Dj' are assigned according to (3) and those in Di'' and Dj'' are given according to (4).

Proof: Let dk0 = log [P(dik= 0 | R)/P(dik= 0 | I)]

Note that dk0 is independent of i, i.e. the value of dk0 does not depend on the document where term k has frequency occurrence = 0.

$$f(Q,Di'') - f(Q,Dj'')$$

$$= \sum_{k=1}^{n} qk(dik' - dk0) - \sum_{k=1}^{n} qk(djk' - dk0)$$

$$= \sum_{k=1}^{n} qk\, dik' - \sum_{k=1}^{n} qk\, djk'$$

$$= f(Q,Di') - f(Q,Dj')$$

4. EXPERIMENTAL RESULTS:

Experiments are carried out for 2 collections: the CRN4NUL collection of 424 documents and 155 queries and the MEDNUL collection of 450 documents and 30 queries. The average (average over the recall points 0.0, 0.25, 0.5, 0.75, 1.0) performance improvements of the non-binary independence model over the binary independence model are 57% and 23% respectively for the two collections. Detailed results are given in tables 1 and 2.

5. CONCLUSION:

Optimal retrieval using non-binary frequency information is given.Experimental results on two collections indicate that very significant improvement over the binary independence model is obtained. In contrast, the tree dependence model [VanR] provides only slight improvement over the binary independence model when the terms used are restricted to the original query terms [p.196, HaVa]. (It should be pointed out that the collections used in the two experiments are different.) Thus, if major improvement in retrieval effectiveness is desired, term frequency information should be utilized. We will be seeking appropriate assignment of term weights when relevance information is not available.

The assumption that terms are independent is not entirely unrealistic in view of the fact that there is an effective procedure that can transform a set of terms into a new set of terms that are independent [Wong]. We believe that a proper transformation of terms into independ-terms followed by appropriate incoporation of non-binary information into the retrieval process will yield excellent results.

6. REFERENCES:

[BoSw] Bookstein A. and Swanson D.R. "Probabilistic models for automatic indexing", Journal of the American Society for Information Science, Vol. 25, 1974, pp. 312-319.

[Croft] Croft, W.B. "Experiments with representation in a document system", Techniacl report 82-21, University of Massachusetts, Amherst, Mass. 1982.

[HaVa] Harper D. and Van Rijsbergen C.J. "An evaluation of feedback in document retrieval using co-occurrence data", Journal of Documentation, Sept. 1978, pp. 189-216.

[Hart] Harter, S.P. "A probabilistic approach to automatic keyword indexing", Journal of the American Society for Information Science, Vol. 26 1975, part I. pp. 197-205, part II. pp. 280-289.

[Lose] Losee, R. "The performance of probabilistic models of document re trieval systems", Ph.D. thesis, University of Chicago (to appear in 1986)

[RaSY] Raghavan, V., Shi, H. and Yu, C. "Evaluation of the 2-Poisson models as a basis for using term frequency data in searching", ACMSIGIR conference, 1983.

[Robe] Robertson, S.E. "The probability ranking principle in information retrieval", Journal of Documentation, 1977, pp. 294-304.

[RoSp] Robertson, S.E. and Sparch Jones, K. "Relevance weighing of search terms", Journal of the ASIS, 1976, pp. 129-146.

[Wong] Wong, M., Ziarko, W. and Wong, T.C. "Generalized vector space model in information retrieval", ACM SIGIR 1985, pp. 18-25.

[VanR] Van Rijsbergen, C.J. "A theoretical basis for the use of cooccurrence data in information retrieval", Journal of Documentation, 1977, pp. 106-119.

[YuSa] Yu, C.T. and Salton, G. "Precision weighing - an effective automatic indexing method", J. ACM, 1976, pp 76-88

	precision	
Recall value	Binary independence	Non-binary independence
0.000	0.831	0.981
0.050	0.826	0.981
0.100	0.814	0.979
0.150	0.772	0.966
0.200	0.762	0.964
0.250	0.720	0.961
0.300	0.663	0.940
0.350	0.598	0.913
0.400	0.586	0.905
0.450	0.551	0.848
0.500	0.547	0.842
0.550	0.475	0.770
0.600	0.460	0.756
0.650	0.436	0.732
0.700	0.367	0.665
0.750	0.350	0.637
0.800	0.289	0.549
0.850	0.242	0.489
0.900	0.222	0.451
0.950	0.213	0.432
1.000	0.213	0.432

Table: 1 (CRN4NUL)

	precision	
Recall Value	Binary independence	Non-binary independence
0.000	0.856	0.966
0.050	0.856	0.966
0.099	0.856	0.966
0.150	0.819	0.966
0.200	0.809	0.945
0.250	0.785	0.941
0.300	0.761	0.884
0.350	0.754	0.877
0.400	0.753	0.876
0.450	0.724	0.862
0.500	0.716	0.841
0.550	0.671	0.801
0.600	0.656	0.780
0.650	0.629	0.770
0.700	0.622	0.744
0.750	0.589	0.735
0.800	0.477	0.612
0.850	0.437	0.554
0.900	0.362	0.455
0.950	0.259	0.360
1.000	0.259	0.360

Table: 2 (MEDNUL)

THE MAXIMUM ENTROPY PRINCIPLE IN INFORMATION RETRIEVAL*

Paul B. Kantor & Jung Jin Lee
Tantalus, Inc., Cleveland, Ohio

ABSTRACT

Applications, assumptions and properties of the maximum entropy principle are discussed. The maximum entropy principle integrates prior estimates of relevance with the observed distribution of term combinations. The result may be a reordering of the segments of a database, compared to a naive estimate. Numerical examples obtained by solution of the non-linear equations for the dual variables are presented and discussed.

* Supported in part by the National Science Foundation under grant IST-8318630.

BACKGROUND

It is hardly necessary to state the importance of information retrieval. Whether for research, business or military purposes, the speedy and accurate retrieval of relevant information can mean the difference between success and failure. Information retrieval (IR) as a discipline has been primarily developed for the storage and retrieval of text materials. In the early days, it was easy to scoff at the emphasis given by research to the specific textual content of a document. [CALV81] However, with the onslaught of digital electronic storage of documents, it becomes clear that this emphasis was prophetic. All of the familiar clues -- size, shape, location within an office or a file cabinet and so forth are being taken away from us. Our ability to retrieve information quickly and accurately hinges on our ability to describe the items to be retrieved in terms of their textual content (and perhaps a few dozen additional descriptors).

The general outlines of term description of documents are given in Salton's monograph [Salt83]. Other kinds of application are becoming increasingly important. An example of non text information retrieval is provided by a network of weather observation stations. Each station compiles an extensive file of data on wind direction and speed, barometric pressure, precipitation and other meteorological data. It reports, to a central coordinating agency or computer, only a few summary statistics. These may be thought of as abstracts, or "key terms" describing the full file of information available at the weather station.

If the problem of weather prediction is to be handled in real time, it appears essential for the central agency or computer to be able to determine, from those abstracts, which detailed reports are essential to resolve ambiguities in short and medium range forecasting. The problem of coordinating the enormous quantity of available data is not likely to be solved by building a computer that can digest all of it at once. It is much more likely to be solved by a disaggregated scheme in which the central computer knows how to process abstracts, and has effective rules for determining when to request the "full document" from a reporting station.

The same hierarchical structure is present, with much more stringent real time constraints, in the design of a national defense system. Such a system, as presently conceived, would integrate vast quantities of data from observation stations, to estimate whether or not a threat is present, and if so to take appropriate responsive action. Again, it seems unlikely that this problem will be solved by simultaneous processing of all the available data. A much more likely solution path is the development of one or several centralized units, each capable of evaluating abstracts in real time and requesting appropriate further information.

What may also mention an intriguing relation to the field of expert systems (ES), and its more all-encompassing cousin, artificial intelligence (AI). It is widely recognized that both ES and AI have much to contribute to the area of document retrieval (GUID83). Expert systems can handle the complexity of dealing with a variety of databases and retrieval languages, once a query is cast into some standard "search language". Artificial intelligence, in its turn, can work to process natural language input, to reduce an unsophisticated expression of information need to a suitable standard form. What is less widely recognized is that information retrieval itself has much to contribute to both ES and AI.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

Information retrieval is important in ES and AI because any nontrivial expert system, or intelligent program, must process new information or queries in terms of a substantial base of facts and rules. For all but the simplest queries, repeated scanning of the base of facts and rules can consume an enormous amount of time. If, on the other hand, the central processor is able to retrieve those facts and rules must likely to contribute to resolution of the query, the whole process will be enormously speeded. An initial acceleration can be achieved simply by a thorough inverted indexing of the base of facts and rules. However, the term indexing, by the logical equivalent of "document descriptors" is likely to offer substantial improvements.

In the present report, no specific application is made of the techniques discussed, but their general properties and their relation to some previous schemes is explored.

THE MAXIMUM ENTROPY PRINCIPLE

COOPER [COOP83] and COOPER and HUIZINGA [COOP82] have proposed that the maximum entropy principle can serve as a powerful tool to improve retrieval of information. The maximum entropy principle was proposed, as a means for unifying a number of physical problems, by JAYNES [JAYN57]. Independently, GOOD [GOOD63] develop maximum entropy as a technique for generalizing the notion of a "null hypothesis" from two dimensional contingency tables, to higher dimensional problems. An extensive discussion on the justification of maximum entropy methods in engineering has been given by JAYNES[JAYN82]. A technical discussion of the formulation of the maximum entropy problem for information retrieval, has been given by KANTOR [KANT84]. The present paper builds upon that reference, to which the reader is referred for more technical details.

The application of the maximum entropy principle to information retrieval has a good deal in common with Good's formulation. The entire set of "documents" (we shall use the term "documents" to refer in general to any retrievable report or item that might contain further information relevant to the problem at hand.) consist of items to which any of a number of descriptors may apply. If there are, say, 24 descriptors, one might try to represent this as a 24 dimensional contingency array. The number in each cell of this array would represent the number of documents having a particular combination of descriptions. We want to estimate the "relevance" of the documents in a given cell. This could be expressed by adding one more dimension to the contingency table. In order to express our results on paper, we resort to a two dimensional contingency table in which the rows represent all the cells of the very high dimensional table, and the columns represent the possible degrees of relevance. We call this the Grand Contingency Table.

In the simpliest case, the descriptor terms are given an order, fixed once and for all. The rows can then be labeled by a string of 0's and 1's indicating the absence or appearance of the term in the document. The rows are easily arranged lexicographically. In a more general formulation, the elements of the document vector could be integers greater than one, representing for example the number of times that a term appears in a document. The principle of formulating a two dimensional contingency table still applies. Descriptions could be even more general, such as "appears related to a document with description D". Such descriptions are likely to occur in the relevancy feedback models described by Salton. Although there might be some difficulty in defining a unique order for the rows of the table, this generalization also does not conflict with the essential two dimensionality of the problem.

The columns, as stated, correspond to varying possible degrees of relevance. In the remainder of this paper we shall make the classical assumption that the degree of relevance is either 0 or 1. However, we shall speak of the "expected relevance" of a document drawn from the set described by a particular row. This "expected relevance" is simply the number of relevant documents in that set divided by the total number of documents in that set. The notion of expected relevance can be easily generalized to a case where relevance assumes several discrete values, and even to the case where it may be continuously distributed between 0 and 1.

Stated in its simplest terms, the maximum entropy principle makes the following assertions:

(1) The distribution of documents into the rows of this table is given by the database. It is a fact, not open to speculation, and this distribution need not obey any specific assumptions.

(2) Some prior information on the expected distribution of relevant documents in this large table is available to us. In the most natural case, which will be used throughout this discussion, this information is given in terms of the probability that an item characterized by an individual descriptor (D_j) will be relevant. (It is common in retrieval systems to form the Boolean union of several terms expected to be nearly synonymous. This does not provide an essential complication as, in what follows, the word "descriptor" can also be interpreted to mean "union of closely related descriptors.")

(3) Finally, the relevant and non-relevant documents are assumed to be distributed, within this grand contingency table, as randomly as possible subject to the constraints of (1) and (2).

A constraint of the type (2) tells something not about a single row of the grand contingency table, but about half of the rows together -- that half to which the description (D_j) applies. Of course, the goal of an efficient retrieval system is to estimate which of the specific rows have the highest expected relevance.

As JAYNES and GOOD and KHINCHIN [KHIN67] have shown, the only defensible definition of the phrase "as randomly as possible" is "the distribution having the largest entropy."

The entropy function is defined up to a multiplicative constant by equation 1.

Eq. 1 $S(p_1,p_2,\ldots p_{2.2^N}) = -\sum_i p_i \ln p_i$

The choice of the base of logarithms is arbitrary, and for optimization analysis, it is most convenient to use logarithms to the base The resulting constrained optimization problem is to maximize the entropy function subject to a set of limited constraints of the form given in equation 2.

Eq. 2 $\sum_{R\epsilon R(I)} n(R)v(R) = V_I \sum_{R\epsilon R(I)} n(R)$

The terms in this equation are defined as follows:

R *labels rows.* R=(00000.., 10000..., ..., 111111...)
I *labels constraints* $I = 1, \ldots k$
$R(I)$ = R *such that constraint* I *applies to* R
V_I = *expected relevance as in assumption (3)*
$v(R)$ = *expected relevance of row* R

Eq. 3

It can be shown (Kantor 1984) that if this problem has a solution, the solution may be expressed in terms of a set of Lagrange multipliers $\lambda_1,\ldots\lambda_k$, where k is the number of constraints provided. It is further shown that the richness of any row is given by Equation 4.

Eq. 4. $v(R) = v(\sum_{I\epsilon C(R)} \lambda_I)$

In this equation the set $C(R)$ is defined as in equation 5, and the universal value function v is defined as in equation 6.

Eq. 5 $C(R) =$ {I *such that* I *constrains* R}

Eq. 6 $v(x) = e^x/(1+e^x)$

The particular form of the function is determined by our assumption that relevance assumes only the values 0 and 1. For a more general discussions see [KANT84].

When Equation 4 is inserted into Equation 2, a set of coupled equations results. The number of these equations is equal to the number of independent constraints that have been provided. The equations are given in Equation 7

Eq.7 $\sum_{R\epsilon R(I)} n(R)v(\sum_{J\epsilon C(R)} \lambda_J) = V_I \sum_{R\epsilon R(I)} n(R)$

These equations are non-linear, and do not always have a real solution. When they do have a solution, the values of λ_J are determined by the constraint values V_I , AND the actual distribution of items among the rows of the grand contingency table $\{n(R)\}$ The additional dependence on the actual distribution can produce surprising and significant results. In the remainder of this paper, to keep the discussion within bounds, we will consider only the case of two descriptive terms.

RELATIONSHIP TO TERM INDEPENDENCE

Suppose we know a partial table giving the distribution of relevant and non-relevant items in a very small data base. The numbers are as shown in Table 1.

Table 1. A hypothetical distribution

Row(R)	n(R)	not-Rel	Rel	z(R)
00	84	72	12	12:72
10	46	18	28	28:18
01	26	8	18	18:8
11				

In each row we show the total number of documents having the corresponding description, the number that are not relevant, the number that are relevant, and the odds ratio for relevance as apposed to not-relevance. For example, for items described by the first descriptor alone the odds ratio is 28 to 18. With reference to Equation 4 we see that the logarithm of the odds ratio is related to the Lagrange multipliers as shown in Equation 8.

Eq. 8 $\ln z(R) = \sum_{I\epsilon C(R)} \lambda_I$

The last row of this table has been left empty so that we may easily contrast the concept of binary term independence with two formulations of the Maximum Entropy principle

The principle of binary independence requires that the last row be as shown in Table 2.

Table 2. The completion under Binary Independence

Row (R)	n(R)	not-Rel	Rel	z(R)
11	44	2	42	42:2

The odds ratio for the appearence of both terms is the "natural" combination of the odds ratios for the three preceding terms as shown in Equation 9.

Eq. 9 $z(11) = z(10)z(01)/z(00)$

This equation states that addition of either term enhances the other in the same way that presence of the added term is an enhancement compared to the absence of either term. This result is plausible, and deserves to be preserved.

On the other hand, the binary independence model has also insisted that there be exactly 44 elements in the row 11. This insistence may or may not be a true description of the data base, and violates principle (1) of the maximum entropy formulation. It is not hard to see that in the extention to three or more descriptors the degree to which the problem is overconstrained increases rapidly.

TWO FORMULATIONS OF THE MAXIMUM ENTROPY PRINCIPLE

The maximum entropy principle may be applied to this problem in two different ways. In one way no prior assumption is made about the fraction of items (in the entire data base) that are relevant to the problem at hand. The resulting final row is shown in Table 3.

Table 3. Completion By the Minimal MEP

Row (R)	n(R)	not-Rel	Rel	z(R)
11	---	---	---	28:8

The odds ratio is substantially lower than in the binary independence model, but on the other hand no constraint is placed on the total number of items in this row.

A second formulation (which is the form first given by Cooper) supposes that there is some prior estimate of the probability that any item in the data base will be relevant. In this case the final row is as shown in Table 4.

Table 4. Single constraint added to the MEP

Row (R)	n(R)	not-Rel	Rel	z(R)
11	---	---	---	42:2

Note that the number of elements in the row again remains unconstrained, but the attractive rule for combination of odds ratios is preserved from the binary independence model. Thus, the transition to the maximum entropy principle, in this case loses nothing, except the unreasonable constraint that the total number of items described by both terms together be 44.

It should be noted that recent work by SARACEVIC and KANTOR (1986) (unpublished) provides a data base of some 6,000 retrieved items of known relevance for a variety of realistic research problems. Thus it is possible to test the odds ratios predicted by either of the maximum entropy models, and the distribution prediction of the binary independence model, against a substantial set of real data.

HOW DOES ALL THIS COME ABOUT?

The expression given as Equation 7 may be specialized to the two cases described above. In the case where there is no overall constraint on the items in the data base, there are two unknown multipliers and two coupled equations (Equation 10).

Eq. 10a $n(10)v(\lambda_1)+n(11)v(\lambda_1+\lambda_2) = V_1[n(10)+n(11)]$

Eq. 10b $n(01)v(\lambda_2)+n(11)v(\lambda_1+\lambda_2) = V_2[n(01)+n(11)]$

When there is also a constraint on the expected relevance of a document chosen entirely at random, there is one additional Lagrange multiplier λ_0, and correspondingly an additional equation.

Eq. 11a $n(10)v(\lambda_0+\lambda_1)+n(11)v(\lambda_0+\lambda_1+\lambda_2) = V_1[n(10)+n(11)]$

Eq. 11b $n(01)v(\lambda_0+\lambda_2)+n(11)v(\lambda_0+\lambda_1+\lambda_2) = V_2[n(01)+n(11)]$

Eq. 11c $n(00)v(\lambda_0)+n(10)v(\lambda_0+\lambda_1)+n(01)v(\lambda_0+\lambda_2)+ n(11)v(\lambda_0+\lambda_1+\lambda_2)=V_0[n(00)+n(10)+n(01)+n(11)]$

It is important to realize that the prior estimates V_0, V_1, V_2 represent "naive" estimates. The computed values of the Lagrange multipliers $\lambda_1, \lambda_2, \lambda_0$ are what determine the odds of relevance. The transformation between them is not simple. It depends in an essential way on the actual distribution of terms among items (or in a language more natural to our Grand Contingency Table, the distribution of items among term combinations.) This dependence is not surprising, since one key motivation for the maximum entropy principle is to use the information provided by the given distribution of terms.

The equations need not always have solutions. If the row sizes are as given in Equation 12

Eq. 12 $n(10) = 10$, $n(01) = 10$, $n(11) = 80$

then the initial conditions V_1=90%; V_2=10% are impossible. The first of these conditions requires that 71 of the 80 in n(11) be relevant. The second condition requires that 71 of those same items be not relevant.

Any neutral point at which all of the naive estimates are equal can always be solved. We use the neutral point with each estimate equal to 0.5 (the "vapid point") as the starting point in more difficult problems with higher dimensionality.

An example of the impact of the $\{n(R)\}$ in changing the rank order of the Lagrange multipliers is given in Table 5

We have not found an intuitive characterization of the permutations that occur. At this point it appears that any reordering of the naive estimates of relevance is possible in a suitably distributed data base.

SIMULTANEITY VS PATHWISE ANALYSIS

The shortcomings of term independence have been known for some time. [CHEES83], [Yu83]). The problem can be attacked by a pathwise approach to term combinations. These approaches can become quite complicated. They bear some similarity to the stepwise regression techniques used in linear modeling. Extending the same analogy, the maximum entropy approach is like multiple regression in that it deals with all of the information simultaneously. It would seem, whenever it can be implemented, that this principle is to be prefered.

Using available state of the art optimization programs (not to maximize the entropy, but to solve the coupled nonlinear equations) we have been able to handle solveable problems with 5 independent constraints in times of the order of 10 seconds, running on an IBM-PC compatible micro computer with an 8087 math chip, at a low clock speed.

TWO INTERPRETATIONS OF "FUZZINESS" OR "WEIGHT"

In Equation 11 we may introduce "weight" parameters in two different ways. Both of these arise from reinterpreting the incidence matrix defining the relationship between constraints and rows. This incidence matrix is shown as Table 6.

Table 6: The incidence matrix

	1	2	0
00			✓
10	✓		✓
01		✓	✓
11	✓	✓	✓

A weight can be introduced into this matrix by interpretting each check mark as a 1 and changing one or more of the check marks to numbers less than 1. This is shown in Table 7.

Table 7: Weighted incidence

	1	2	0
00			1
10	g		1
01		1	1
11	1	1	1

In words, this transformation may be expressed as "we are not so sure about the relevance of term 1 when it occurs alone without term 2". The modifications effect only Equation 11a, which then takes the form shown in Equation 13.

Eq. 13. $$gn(10)v(\lambda_0+\lambda_1)+n(11)v(\lambda_0+\lambda_1+\lambda_2) = V_1\left[gn(10)+n(11)\right]$$

Another possible interpretation, which appears not to be equivalent to the first is shown in Equation 14.

Eq. 14. $$n(10)v(\lambda_0+g\lambda_1)+n(11)v(\lambda_0+\lambda_1+\lambda 2) = V_1\left[n(10)+n(11)\right]$$

Conceivably, two independent weight parameters g,h could be introduced simultaneously as in Equation 15.

Eq. 15. $$gn(10)v(\lambda_0+h\lambda_1)+n(11)v(\lambda_0+\lambda_1+\lambda_2) = V_1\left[gn(10)+n(11)\right]$$

The implications of this concept remain to be explored.

GENERALIZATION TO CONTINUOUS VALUE.

If value is not assumed to take only the values 0 and 1 but can assume several values in the unit interval, the effect is to build up a conditional entropy maximization problem. This is equivalent to a minimum cross-entropy model, with a prior distribution corresponding to 1/n at each of n allowed points in the unit interval. In this language, our present model has a Bayesian prior distribution (0.5, 0.5).

An accumulation of points can be used to approach a continuous distribution. In another point of view, the value may be assumed to vary continuously in the unit interval, with a uniform prior distribution. This case has been considered by KANTOR [KANT84]. The structure is essentially the same, and leads to Equation 7. However the definition of the universal value function is changed, to the Langevin function, Equation 16. The Langevin function is known from classical statistical mechanics.

Eq. 16. $$v(\lambda) = 0.5 + 0.5\left[\coth \lambda - 1/\lambda\right]$$

In this model, which is a more realistic way to consider relevance, the odds ratio will not be simply multiplicative. The full implications of this model have not yet been explored.

Table 5. Examples of the change in ordering of intersection sets, induced by changes in the composition of the data base. This example is for three distinct terms, labeled A,B,C. The entire data base is refered to as R. Thus V1=0.1; V2= 0.2; V3= 0.2 and V0=0.5 in the first example. In the last 8 columns the Boolean atoms are shown in order of increasing richness.

V(A) = 0.100 V(B) = 0.200 V(C) = 0.200 V(R) = 0.500

n(000)	n(C)	n(B)	n(BC)	n(A)	n(AC)	n(AB)	n(ABC)	OBJ F.							RANK
0.50	0.10	0.10	0.05	0.10	0.05	0.05	0.05	0.0000	ABC	AC	AB	BC	A	C	B 000
0.50	0.05	0.05	0.10	0.05	0.10	0.10	0.05	0.0000	ABC	AC	AB	A	BC	C	B 000
0.30	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.0000	ABC	AB	AC	BC	A	B	C 000
0.30	0.05	0.05	0.10	0.05	0.10	0.10	0.25	0.0000	ABC	AB	AC	BC	A	B	C 000
0.10	0.10	0.10	0.20	0.20	0.10	0.10	0.10	0.0090	ABC	AB	AC	BC	A	B	C 000
0.10	0.10	0.05	0.30	0.30	0.05	0.05	0.05	0.0145	ABC	AB	AC	BC	A	B	C 000

V(A) = 0.600 V(B) = 0.800 V(C) = 0.500 V(R) = 0.100

n(000)	n(C)	n(B)	n(BC)	n(A)	n(AC)	n(AB)	n(ABC)	OBJ F.							RANK
0.50	0.10	0.10	0.05	0.10	0.05	0.05	0.05	0.0079	000	C	A	B	AC	BC	AB ABC
0.50	0.05	0.05	0.10	0.05	0.10	0.10	0.05	0.0147	000	C	A	B	AC	BC	AB ABC
0.30	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.0310	000	C	A	AC	B	BC	AB ABC
0.30	0.05	0.05	0.10	0.05	0.10	0.10	0.25	0.0467	000	C	A	B	AC	BC	AB ABC
0.10	0.10	0.10	0.20	0.20	0.10	0.10	0.10	0.0617	000	C	A	AC	B	BC	AB ABC
0.10	0.10	0.05	0.30	0.30	0.05	0.05	0.05	0.0660	000	C	A	AC	B	BC	AB ABC

V(A) = 0.300 V(B) = 0.500 V(C) = 0.400 V(R) = 0.600

n(000)	n(C)	n(B)	n(BC)	n(A)	n(AC)	n(AB)	n(ABC)	OBJ F.							RANK
0.50	0.10	0.10	0.05	0.10	0.05	0.05	0.05	0.0000	ABC	AC	AB	A	BC	C	B 000
0.50	0.05	0.05	0.10	0.05	0.10	0.10	0.05	0.0000	AC	ABC	A	AB	C	BC	000 B
0.30	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.0000	ABC	AC	AB	A	BC	C	B 000
0.30	0.05	0.05	0.10	0.05	0.10	0.10	0.25	0.0000	AC	ABC	A	AB	C	BC	000 B
0.10	0.10	0.10	0.20	0.20	0.10	0.10	0.10	0.0008	ABC	AC	AB	BC	A	C	B 000
0.10	0.10	0.05	0.30	0.30	0.05	0.05	0.05	0.0044	ABC	AC	AB	A	BC	C	B 000

ACKNOWLEDGEMENTS

It is a pleasure to acknowledge stimulating discussions with Dr. Richard Blankenbecler, Dr. Frederick Kantor, and the participants at the 1983 and 1985 Working Conferences on the Maximum Entropy Principle presented by Professor Ray Smith, at the University of Wyoming. Support from the Office of Naval Research made it possible for one of the authors (PBK) to attend the 1985 conference.

REFERENCES AND LITERATURE CITED

[CALV81] CALVINO, Italo "Se una notte d'inverno un viaggiatore" (If on a winter's night a traveler), Harcourt Brace Jovanovich, p260, c1981.

[CHEE83] CHEESEMAN, Peter "A method of computing maximum-entropy probability values for expert system", Proceedings of the 3rd Annual Workshop on Bayesian Methods in Applied Statistics; C. Ray Smith (ed). Laramie, WY: University of Wyoming; (1983)

[COOP83] COOPER, W. S. "Exploiting the maximum entropy principle to increase retrieval effectiveness", J. Am. Soc. Inf. Sci., v34n1pp31-39 (1983).

[COOP82] COOPER, W. S. and HUIZINGA, P. "The maximum entropy principle and its application to the design of probabilistic retrieval systems", Inf. Technol: Res. & Dev., v1n2pp99-112 (1982).

[GOOD63] GOOD, I. J. "Maximum entropy for hypothesis formulation, especially for multidimensional contingency tables", Ann. Math. Stat. v34pp911-932 (1963).

[GUID83] GUIDA, Giovanni and TASSO, C. "An Expert Intermediary System for Interactive Document Retrieval, Automatica, v19n6p759-766, November 1983. IR-NLI.

[JAYN82] JAYNES, Edwin T. "On the rational of maximum entropy methods", Proceedings of the IEEE, v70n9, September, 1982.

[KANT84] KANTOR, Paul B. "Maximum Entropy and the Optimal Design of Automated Information Retrieval Systems." Information Technology; v3n2pp88-94 (1984).

[KHIN67] KHINCHIN, A. I. "Foundations of statistical mechanics", Dover, USA (1967)

[SALT83] SALTON, Gerard "Introduction to modern information retrieval", McGraw-Hill, 448p, c(1983).

[YU83] YU, C. T., BUCKLEY, C., LAM, K., SALTON, G., "A generalized term dependence model in information retrieval", Technical report TR 83-543, Department of Computer Science, Cornell University (1983).

AN INTERPRETATION OF INDEX TERM WEIGHTING SCHEMES BASED ON DOCUMENT COMPONENTS

K. L. Kwok

Computer Science Department, Queens College,
City University of New York, Flushing, NY 11367, USA.

Abstract

A theory of indexing is presented and is based on viewing a document as constituted of components. A component may be chosen as any run of text unit that can be: (a) judged as to its relevency property; and (b) considered as independent within the document. By looking at the constituent components of a document in relation to the universe of all components from the collection, we have been able to apply Bayes' decision theory to derive the index term representation for the document, as well as attaching an initial probabilistic weight for each term based on a Principle of Document Self-Recovery. It turns out that different choices of document components, such as a word or a whole abstract, can lead to different term weighting schemes that have been introduced before and are based on probability considerations; specifically, Edmundson and Wyllys' term significance formula, Sparck Jones' inverse document frequency, and later modified by Croft and Harper into the 'combination match' formula. Thus, a unified interpretation of various probabilistic term weighting schemes appears possible.

1. Introduction

In [KWOK85a] we have introduced a new model of viewing a document's content under the circumstances that a set of relevant cited documents are given with the document (which we call the source). That approach is based on employing Bayes' decision theory in trying to recover, with minimum probability of error, the original source and its cited document cluster among the augmented set of all documents (source plus cited). This then leads to a representation for the source which is consisted of: (1) the union of all the index terms of its components; and (2) every term is associated with a probabilistic weight. This has been referred to as indexing with author-provided relevance information [KWOK85b] because the author, when composing the source document, normally provides a list of cited documents that are used to support the source, and which usually have a high concentration of items relevant to it.

Although cited documents are known to be useful for IR in various contexts for a long time, databases containing them are not common (except for those produced by ISI). In the majority of evaluated collections for experimental investigation, (or for any collection in general), the textual part of a document usually contains only the title, or title combined with abstract as surrogate. In this paper we attempt to extend our theory to documents with this kind of textual information only, and to derive their term indexing and weighting accordingly. It seems that this is possible if a document is viewed as consisting of components, each component being independent and can be judged as to its relevancy property. By varying the size or graininess of these document components, we are able to derive some of the popular term weighting schemes that have been proposed before; specifically the contrast ratio of a term's relative frequency within a document to that in the whole collection as introduced by Edmundson and Wyllys [EDMU61], and the inverse document frequency (IDF) introduced by Sparck Jones [SPAR72] and later modified by Croft and Harper [CROF79]. Thus it will be seen that our theory of indexing can provide a unified interpretation of some of the weighting schemes that are based on probability, and that have been introduced in a more or less ad hoc fashion.

In Section 2 we introduce the concept of document components. Section 3 reviews the probabilistic indexing formalism of Maron and Kuhns [MARO60], and discusses the associated initial indexing and other problems. In Section 4 our approach to indexing and its relation to other previous index term weighting schemes are presented. Section 5 presents results of a trial experiment and Section 6 contains the conclusion and further discussions.

Permission to copy without fee all or part of this material is granted provided that the copyright notice of the "Organization of the 1986-ACM Conference on Research and Development in Information Retrieval" and the title of the publication and its date appear.

© 1986 Organization of the 1986-ACM Conference on Research and Development in Information Retrieval

2. Document Components

For our purpose, a document is to be viewed as consisted of components. A document component may be defined loosely as some fairly self-contained unit of text which has the following properties: (a) each is statistically independent of one another within a document; and (b) each can be judged as to its relevancy property by a user (who may be an enquirer or the author). To satisfy property (b), almost any syntactic unit may be regarded as a component; for example, a word, a phrase, a sentence, multiple sentences, a paragraph, a section, etc. The question of how to satisfy property (a), namely, to be able to isolate conceptually independent components within the full text or surrogate of a document is much more difficult, and probably will have to rely on some natural language analysis techniques together with some AI capabilities. For our investigation we would, as usual, assume the independency property and consider the simplest types of component, namely, a word or a sentence or a whole abstract. The resultant formulation or effectiveness of retrieval will then lead us to justify which of our initial choice of components is the more appropriate.

The concept of document components is of course not new. Approaches to IR via linguistic analysis have often tried to isolate phrases as components to represent more precise concepts. Examples are: Salton's criterion trees [SALT68], Hillman's predicate representation of sentences [HILL62], and the more pragmatic attempts of Klingbiel [KLIN73] and Dillon [DILL83] to isolate noun phrases. O'Connor [OCON80] has considered the retrieval of answer passages as relevant components from full text documents. For lengthy documents, reports, books and other composite documents such as those considered by Fox [FOX85] it is also natural to break down an item into components. In these cases, the purpose of the breakdown of a document is to try to isolate units with more specificity in concept and topicality. For our case, we further look to document components for the purpose of index term weighting.

The consideration of a document as constituting of components may also be beneficial for relevance feedback as discussed by Rocchio [ROCC71], and its application to the case of query term weighting during probabilistic retrieval as discussed in Robertson and Sparck Jones [ROBE76]. Current strategy only allows a user to indicate whether a document is relevant or not. Since documents generally can be cross-disiplinary, multi-topical in nature, and involving a large variety of concepts, it is quite possible that a user may be interested in only one part of a document and may care less for some other portions. If during relevance feedback a user is allowed to indicate the components of a document that is actually relevant, then one may result in more specific feedback information. This option seems to be a logical and useful strategy to implement.

For our purpose of document indexing, a set of indexing terms of size t is assumed given for content description. A document d_i is viewed as consisted of o_i independent components. The aggregate of components from all the N documents in the collection then defines a universe of components of size $N_o = \sum_i o_i$. This will be useful for our index term weighting formulation later. A typical component is represented by a binary vector denoting the presence/absence of index terms, e.g.:

$$u_j = (x_{j1},..x_{jk},..x_{jt}), \qquad x_{jk} = 1/0.$$

At this point we make no assumptions as to what the document components may be. In Section 4 we will show that different kinds of components can lead to different kinds of index term weighting schemes.

3. Probabilistic Indexing Formalism of Maron and Kuhns

In the standard probabilistic indexing theory of Maron and Kuhns [MARO60] and dealt with in a unified fashion in Robertson, et.al. [ROBE82], documents and queries are assumed to be represented via a set of indexing terms. The use (query) experience of a document d_i is then accumulated over time and eventually a set of queries relevant to this document is identified, see Fig. 1a. Together with the occurrence characteristics of the document indexing terms in this set of relevant queries, they allow us to estimate the prior probability $P(+R_i)$ that d_i will be found relevant to some query, and the conditional probability $P(q_a/+R_i)$ that, given relevance to document d_i, the pattern of index terms in a query q_a is used. The product of these two then is proportional to the probability that, given the query q_a, it will be found relevant to d_i, namely, $P(+R_i/q_a)$. Because of difficulties in handling dependencies among index terms, the formulation usually assumes that all index terms are independent, which then leads to probabilistic weights defined for the terms of d_i that will provide an optimal matching strategy between a query q_a and the document d_i.

This formalism has many advantages. First, weighting of indexing terms of a document becomes adaptive and can change over time with the current perception of the document content. Moreover, because users provide the feedback, relevancy judgment is need-oriented rather than topic-oriented. This approach is of course analogous to the probabilistic retrieval schemes of Robertson and Sparck Jones [ROBE76] and van Rijsbergen [VANR77], where currently judged relevant documents are used to weight the terms in a query, see Fig. 1b. From a practical point of view, however, the advantages become less obvious. A document that has not been judged relevant to any queries yet (e.g. a new one) could not be weighted (and is equivalent to the initial search problem in probabilistic retrieval). When one looks at real-life queries (not those made up from document extracts), one also finds that they are generally short, terse and most of them not well thought-out. Indeed, the process of formulating queries that contain sufficient and specific descriptors without intermediary help seems to be a major problem in

IR. Casual users usually are at a loss as to what to put in a query; rather they expect the system to 'know' what they want by giving just the barest clues. Because each query is typically short, a large number of them has to be accumulated to provide usable parameter estimates. For example, many terms of a document may not appear in the set of relevant queries. This may mean that one has to wait a long time (to accumulate relevant queries) before good probabilistic indexing term weights for a document can be obtained.

4. Indexing and Term Weighting based on a Principle of Document Self-Recovery

In view of the discussion in the previous Section, our approach is to give up the advantages of adaptive and need-oriented weighting, and to weight the terms of a document in relation to the collection rather than to a set of queries (see Fig. 2a). Implicitely this assumes that the characteristics of term usage in the queries are similar to that in the documents. Because documents are comparatively long and well-composed, the disadvantages of the query-based weighting scheme can be avoided. Moreover, since the documents in a collection are available immediately without time delay, the initial indexing problem can also be overcome, as discussed below. From another point of view, weighting via the document collection may be regarded as using topical-relevancy information, while query-based weighting makes use of need-relevancy information. Of course, after one has accumulated sufficient number of queries one can always revert back to query-based weighting of Section 3 if one so wishes, but at the initial indexing stage, there is no choice.

Since we will consider our documents as composed of components as discussed in Section 2, our universe is now a set of components of size N_o. In the absence of any additional relevancy information, the constituent components of a document d_i are the only known items relevant to it at this initial indexing stage, see Fig. 2b. They are relevant because the author employs them directly to express his/her thoughts on the topics and concepts that are being written on. We will therefore proceed as formulated in [KWOK85a] and assume d_i to be represented by the binary vector $(y_{i1},..y_{ik}..y_{it})$, where y_{ik}=1/0 denotes the presence/absence of term k in d_i, and is unknown and to be determined. The following decision question is then posed: what representation should d_i take, such that when it is compared with all components in the universe with respect to the relevancy property to d_i, it can re-assemble its constituent components with minimum probability of error? This may be called document indexing based on a Principle of Document Self-Recovery, meaning that the document d_i should be indexed and weighted in such a way that currently available relevance information should be satisfied (i.e. re-assembling the relevant component set of d_i) with minimal probability of error relative to the whole collection. This can be decided, using Bayes' decision theory as follows:

If $O(R_i/u_j) > 1$, then decide u_j relevant to d_i, for all j.

Here, $O(R_i/u_j) = P(+R_i/u_j)/P(-R_i/u_j)$ is the odds (ratio of probability of relevance ($+R_i$) to nonrelevance ($-R_i$)) that given a component u_j, it is relevant to document d_i. As usual, by using Bayesian inversion, the decision can be put into the following form:

If $v(d_i,u_j) > T$, then decide u_j relevant to d_i for all j,

where

$v(d_i,u_j) = \log O(u_j/R_i)$, which may be called a relevance value of component u_j to the topics and concepts of d_i; and

$T = -\log O(R_i)$, a threshold, interpreted as the log of prior odds of document d_i found relevant to some component.

Since the threshold T is unknown, the usual practice is to order all components u_j in decreasing order of the relevance value $v(d_i,u_j)$ with respect to d_i the top-ranked components then are regarded relevant. By assuming term independence in the relevant and nonrelevant sets for the representation of each component, we can put the relevance value into a more useful form as follows:

$$v(d_i,u_j) = \Sigma_k\, w_{ik}\, x_{jk} + \text{constant}$$

where

$$w_{ik} = \log \left[\frac{(p_{ik}/(1-p_{ik}))}{((1-q_{ik})/q_{ik})} \right] \qquad (1)$$

and

$$p_{ik} = P(y_{ik}=1/+R_i)$$
$$q_{ik} = P(y_{ik}=1/-R_i)$$

Here, p_{ik}, q_{ik} are the conditional probabilities that given relevance or nonrelevance to d_i, that term k will occur. Under this circumstance, recovering d_i as constituted of its own components with minimum probability of error, necessary requires that (1) d_i should be represented by the set union of all the terms that are present in its components (hence the vector $(y_{i1}..y_{ik}..y_{it})$ is defined); and (2) that each index term that is present should be weighted by Eq.(1) with p_{ik} and q_{ik} estimated via the actual occurrence of each term among the relevant components of d_i and its complement set in the universe.

Thus, for every indexing term k of document d_i, we will have a 2x2 contingency table as shown in Fig. 3, where

r_{ik} = the number of components within d_i where term k is present;
n_{ok} = the number of components in the universe where term k is present;
c_i, N_o have been defined before.

It remains now to see what forms the weight of each indexing term of d_i would take when we consider the different kinds of components constituting d_i.

4.1 Single Terms as Constituent Components of a Document

The smallest meaningful conceptual component of a document is the single term. If we assume that the index terms in a document constitute its independent components (which we know is not quite correct, because dependencies among terms do exist, see [VANR77], [YU83]), the variables in the 2x2 table of Fig. 3 will have the following specialized meanings:

c_i = L_i, the length or number of terms in d_i, including repeating terms;
r_{ik} = d_{ik}, the term frequency of term k in d_i;
n_{ck} = F_k, the collection frequency of term k = $\sum_i d_{ik}$;
N_c = N_w, the total number of terms used in the whole collection, including repeating terms.

The 2x2 table now appears as in Fig. 4a, and the weight of term k for document d_i becomes:

$$w_{ik} = \log \left[\frac{(d_{ik}/(L_i-d_{ik})) * ((N_w-F_k-L_i+d_{ik})/(F_k-d_{ik}))}{} \right] \quad (2)$$

Edmundson and Wyllys [EDMU61] has suggested using the contrast between a term's relative frequency within a document (viz. d_{ik}/L_i) and its relative frequency in the collection (viz. F_k/N_w) as an indication of a term's significance for content description. One of their suggested formulae is the following:

$$s_{ik} = \log \left[(d_{ik}/L_i) * (N_w/F_k) \right] \quad (3)$$

Thus, it is seen that apart from some second order factors, Eqs.(2) and (3) are the same. In fact, our Eq.(2) is based on the so-called Independence Assumption I2 and Order Principle O2 of Robertson and Sparck Jones [ROBE76]. (I2 assumes that index terms are distributed independently in the relevant and the non-relevant components separately; O2 means that probable relevance is based on both the presence and absence of terms in the components.) Eq.(3), from our point of view however, is based on Independence Assumption I1 (that terms are independent in the relevant set and in the universe), and Order Principle O1 (that probable relevance is based on presence of terms only). It has been noted in [ROBE76] that I2 and O2 are more realistic and have been shown to be more effective than I1 and O1.

Edmundson and Wyllys [EDMU61] actually suggested using the term significance factor as a means for term selection, i.e. terms are ranked according to significance value and those having values higher than a threshold are selected as index terms for a document, see for example Carroll and Roeloffs [CARR69]. However, term significance may also be interpreted as term weight, assuming that we take all terms from the document as indexing terms. They also suggested other functions of (d_{ik}/L_i) and (F_k/N_w) as measures of term significance.

Thus, it can be seen that Edmundson and Wyllys' scheme of term significance is derivable via our approach, and has its basis in assuming that the constituent components of a document are single terms -- a fine-grain approximation.

4.2 Whole Document as Constituent Component of a Document

In the other extreme, the largest meaningful conceptual component of a document is the whole document itself. This would mean that each document describes a very pure and sharp concept, which in reality is not quite correct either. In this circumstance, the variables in the 2x2 table of Fig. 3 specialize to the following:

c_i = 1, meaning the document is its single component;
r_{ik} = 1, terms are present or absent in that component;
n_{ck} = n_k, the document frequency of term k;
N_c = N, the total number of documents in the collection.

For term k, the 2x2 table then appears as in Fig. 4b, and its weight in document d_i becomes:

$$w_{ik} = \log (1/0) + \log (N-n_k)/(n_k-1) = C + \log (N-n_k)/(n_k-1) \quad (4)$$

Here C is a large constant (in theory) corresponding to $p_{ik} \rightarrow 1$. The logarithm term in Eq.(4) is essentially the IDF weighting introduced by Sparck Jones [SPAR72] as:

$$IDF_k = \log (N/n_k) \quad (5)$$

The whole expression in Eq.(4) is, apart from the -1 factor, the same as that introduced by Croft and Harper [CROF79] as:

$$Comb_k = C + \log (N - n_k)/n_k \quad (6)$$

In their formulation, they attached the weight to query terms rather than document terms because they were investigating probabilistic retrieval (Model 2 of Robertson et.al. [ROBE82]). They also considered the similarity measure as the vector product between a binary-weighted document and a query weighted as in Eq.(6); hence the name 'Combinaton Match', meaning that the constant C factor behaves like term coordination, and the second term behaves like IDF.

Thus it can be seen that the Combination Match weighting scheme is also derivable based on our approach to indexing, and has its basis in viewing the whole document as a single component -- a coarse grain approximation.

5. Results of a Trial Experiment

Since the formulae derived in Section 4 differ in their details from those that have been used before, we have carried out a trial experiment using the ADI 82 document collection with 35 queries. This is a small collection and results should be viewed with caution. They are for qualitative illustration purposes, and significance tests of differences among formulae were not calculated. We hope to perform evaluations on larger document collections in the future.

In Fig. 5 we have tabulated the precision values averaged over all 35 queries, at 10 recall levels. The methods for similarity measure between a document and a query are explained below. In what follows, a document $d_i = (d_{i1},..d_{ik},.. d_{it})$ may also be represented as $(y_{i1},..y_{ik},.. y_{it})$ with y_{ik} =1/0, when we ignore the term frequency information. Similarly, a query $q_a = (q_{a1},..q_{ak},.. q_{at})$ may also be denoted as $(z_{a1},..z_{ak},.. z_{at})$ with z_{ak} =1/0 for binary representation.

COO = coordinate matching:
$\Sigma\, COO_k = \Sigma\, y_{ik}z_{ak}$;

COS = cosine formula: $\Sigma\, d_{ik}q_{ak}/\ (/d//q/)$;

IDF = Inverse Document Frequency weighting (see Eq.(5)):
$\Sigma \log\,[\max(n_k)/n_k]COO_k$;

CH = Combination Match formula (see Eq.(6)):
$\Sigma\,[C + \log((N-n_k)/n_k]COO_k$;

CHK = our Eq.(4):
$\Sigma\,[C + \log((N-n_k)/(n_k-1)]COO_k$;

EW = Edmundson and Wyllys' Eq.(3):
$\Sigma \log[(d_{ik}N_w)/(L_iF_k)]COO_k$;

EWK = our Eq.(2):
$\Sigma \log[(d_{ik}/(L_i-d_{ik}))*$
$((N_w-F_k-L_i+d_{ik})/(F_k-d_{ik}))\ ]\ COO_k$;

C = Croft's formula [CROF83]:
$\Sigma\,[K+(1-K)d_{ik}/\max(d_{ik})]\ *$
$[C + \log((N-n_k)/n_k]COO_k$;
(0<K<1, is a parameter to be determined).

CK = Replacing the combination match factor in Croft's formula by our Eq.(4):
$\Sigma\,[K+(1-K)d_{ik}/\max(d_{ik})]\ *$
$[C + \log((N-n_k)/(n_k-1)]COO_k$.

Among the first three formulae, we confirm that, in general for most cases, COO, COS, IDF successively give better performance in this order, as noted by others.

Between CH and CHK, results show that CHK performs better at low recalls and approximately the same or slightly worse at high recalls. In both cases, the constant C took the value ≈ 0, corresponding to $p_{ik} \dashrightarrow 1/2$. As noted by Croft and Harper [CROF79], free text indexing terms derived from queries and documents may not be sufficiently specific to warrant a larger values of p_{ik}; in fact we have observed that their larger values can lead to worse results. On the contrary, some values of $p_{ik} < 1/2$ can lead to improvements of a few percent, but their specific values require determination by trial and error for each formula; hence we choose to use a fixed value of 1/2. In the formula for CHK, the factor n_k-1 is replaced by n_k-1+delta, where delta = 0.05, to take care of some degenerate terms. Results do not seem to change materially on the choice of delta around this value.

The three formulae CH, IDF, CHK are very similar to each other because of the choice of $p_{ik} \dashrightarrow 1/2$. Results show that they perform better successively in this order in low recall regions. For high recalls, they perform about equal or slightly worse successively, in the order IDF, CH, CHK.

Comparing EW and EWK, we see that EWK performs better at low recalls, and the reverse is true for high recalls. The uniformly better result of I2+02 (independence assumption 2, order principle 2) versus I1+01 (independence assumption 1, order principle 1) as noted in [ROBE76], is not observed here, perhaps because the collecion is too small.

EWK performed better than CH, IDF, CHK almost throughout, except for the recall level of 0.2.

Finally, Croft's formula [CROF83] denoted by C, including a factor of term significance $[K+(1-K)d_{ik}/\max(d_{ik})]$ is given. Its modification, denoted by CK, by our new Eq.(4) in place of the combination match factor is also studied. In general they are better than others, except for EW and EWK which give better results at high recall regions. Between C and CK, we observe that they behave about the same or slightly in favor of CK at low recalls, and vice versa at high recalls. These formulae (C and CK) require estimation of an additional parameter K, which we set at 1/2. Other values of K in general give worse results.

Within the context of this small trial experiment, results seem to give the impression that roughly there are two groups of probabilistic weights. The first consists of EW, EWK and C, CK, all having approximately similar performance. A fine detail is that C, CK perform better at low recalls, and the other two better at high recalls. The other group consists of CH, IDF and CHK, and they in general perform less well than the first group. Since CHK is based on the assumption that each document has one single component which is itself, while EWK is based on each word being an independent component, the latter choice of component appears to be more accurate for our theory. Both of these assumptions are not quite correct in reality, and a better choice of component type could be something in-between, such as a phrase or sentence. In fact, the formulation of C, which gives overall results better than

others, appears to correspond approximately to the case when a component is assumed to be a sentence (see Discussion in Section 6). It seems that larger scale experimentation on these term weighting schemes is warranted.

6. Conclusion and Discussion

A unified theory of document indexing based on probability theory, the concept of document components and a Principle of Document Self-Recovery has been presented. It is found that different index term weighting schemes in use actually correspond to different choices of the graininess of the components of a document within our theory. Since methods to determine independent components are not readily available, we would use retrieval effectiveness of the different weighting schemes to determine which choice of component size is the most appropriate. There does not seem to be any published results for Eq.(3) as term weights; however, our limited experiment with the ADI collection of 82 documents shows that it and our modified form, namely, Eq.(2) are among the best weighting schemes available for retrieval.

Currently the best published probabilistic term weighting method seems to be that of Croft's [CROF83], which uses the Combination Match formula multiplied by a term significance factor that is interpreted as the probability that a term will be used as an indexing term for the document. When our theory of indexing is extended to queries, and when a fully normalized symmetric probabilistic similarity measure (similar to that in [KWOK85a]) is used between a document and a query, it seems that Croft's formula may also be approximated based on our approach. In that case the choice of component is found to be intermediate between the two extremes that we have discussed here; for example, a sentence. A more complete paper is being prepared to present the three cases together.

Acknowledgement

The author would like to thank Mr. William Kuan for his help in program development for the experiment. This work was done during the author's receipt of a Queens College Faculty-in-Residence award, and was partially supported by a grant from the Professional Staff Congress of CUNY (PSCUNY 6-65238). The database was kindly provided by Prof. Salton of Cornell University.

References

[CARR69] Carroll, J.M.; Roeloffs, R. 'Computer selection of keywords using word-frequency analysis.' Am. Doc. 20:227-233; 1969.

[CROF79] Croft, W.B.; Harper, D.J. 'Using probabilistic models of document retrieval without relevance information.' J. of Doc. 35:285; 1979.

[CROF83] Croft, W.B. 'Experiments with representation in a document retrieval system.' Info. Tech.: R&D. 2:1-21; 1983.

[DILL83] Dillon, M.; Gray, A.S. 'FASIT: a fully automatic syntactically based indexing system.' J. of ASIS. 34:99-108; 1983.

[EDMU61] Edmundson, H.P.; Wyllys, R.E. 'Automatic abstracting and indexing -- survey and recommendations.' Comm. of ACM. 4:226-334; 1961.

[FOX85] Fox, E. 'Composite document extended retrival -- an overview.' Proc. 8th Ann. Intl. ACM SIGIR Conf. on R&D in IR (Montreal 1985). pp.43-53; 1985.

[HILL68] Hillman, D.J. 'Negotiation of inquiries in an on-line retrieval system.' Info. Stor. Retr. 4:219-238; 1968.

[KLIN73] Klingbiel, P.H. 'A technique for machine-aided indexing.' Info. Stor. Retr. 9:477-494; 1973.

[KWOK85a] Kwok, K.L. 'A probabilistic theory of indexing and similarity measure based on cited and citing documents.' J. of ASIS. 36:342-351; 1985.

[KWOK85b] Kwok, K.L. 'A probabilistic theory of indexing using author-provided relevance information.' Proc. of 48th Ann. Mtg. of ASIS. pp. 59-63. Las Vegas 1985.

[MARO60] Maron, M.E.; Kuhn, L. 'On relevance, probabilistic indexing and Information Retrieval.' J. of ACM. 25:67; 1960.

[OCON80] O'Connor, J. 'Answer-passage retrieval by text searching.' J. of ASIS. 31:227-239; 1980.

[ROCC71] Rocchio, J.J.Jr. 'Relevance feedback in Information Retrieval.' in The SMART Retrieval System -- Experiments in Automatic Document Processing; ed. G. Salton. Prentice-Hall, 1971.

[ROBE76] Robertson, S.E.; Sparck Jones, K. 'Relevance weighting of search terms.' J. of ASIS. 27:129; 1976.

[ROBE82] Robertson, S.E.; Maron, M.E.; Cooper, W.S. 'Probability of relevance: a unification of two competing models for document retrieval.' Info. Tech.: R&D. 1:1; 1982.

[SALT68] Salton, G. Automatic Information Organization and Retrieval, McGraw Hill, 1968.

[SPAR72] Sparck Jones, K. 'A statistical interpretation of term specificity and its application in retrieval.' J. of Doc. 28:11-21; 1972.

[VANR77] van Rijsbergen C.J. 'A theoretical basis for the use of co-occurrence of data in information retrieval.' J. of Doc 33:106; 1977.

[YU82] Yu, C.T.; Buckley, C.; Lam, K.; Salton, G. 'A generlaized term dependence model in information retrieval.' Info. Tech.: R&D. 2:129-154; 1983.

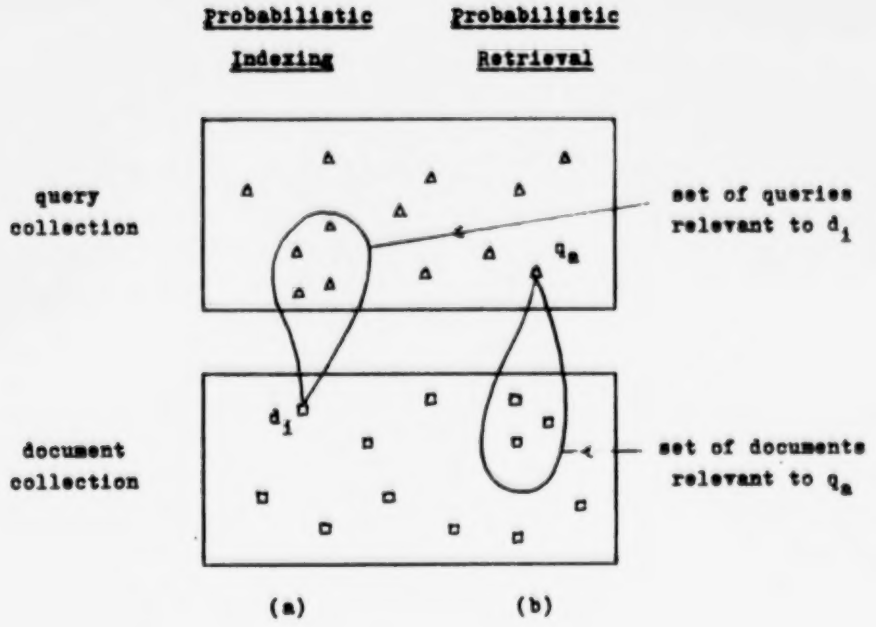


Fig. 1: The Environment of Probabilistic Indexing and Retrieval

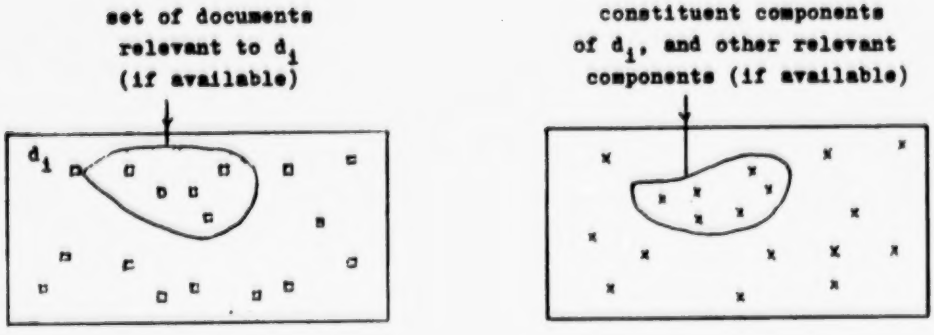

set of documents relevant to d_i (if available)
d_i
constituent components of d_i, and other relevant components (if available)
(a) Document Collection
(b) Universe of Document Components

Fig. 2: Probabilistic Indexing of Document d_i in relation to (a) Document Collection; (b) Universe of Document Components

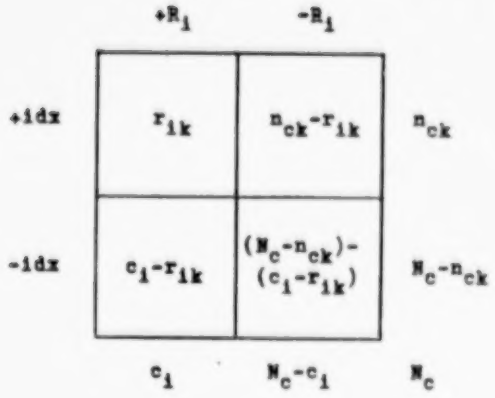


Fig. 3: 2x2 Contingency Table for Term k of Document i in the Universe of Document Components

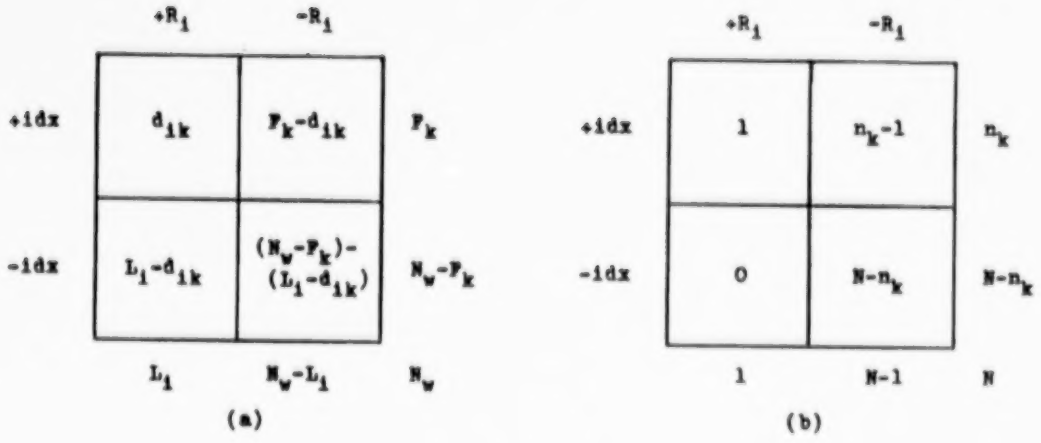


Fig. 4: 2x2 Contingency Table for Specialised Document Components: (a) Single Term; (b) Whole Abstract

Recall	COO	COS	IDF	CH	CHX	EV	EWX	C	CX
0.1	.441	.461	.624	.605	.649	.622	.654	.669	.682
0.2	.397	.434	.615	.600	.644	.610	.636	.658	.659
0.3	.347	.425	.520	.521	.551	.542	.569	.571	.572
0.4	.279	.414	.424	.417	.454	.495	.506	.501	.517
0.5	.269	.385	.414	.410	.447	.467	.480	.497	.512
0.6	.240	.316	.303	.296	.292	.354	.350	.355	.350
0.7	.191	.242	.250	.247	.242	.281	.270	.283	.279
0.8	.179	.230	.228	.226	.221	.269	.258	.258	.249
0.9	.159	.197	.205	.206	.202	.259	.247	.240	.231
1.0	.159	.197	.204	.204	.200	.259	.247	.238	.229

Fig. 5: Precision Values (Averaged over 35 Queries) of the ADI Collection (82 Documents) for 9 Term Weighting Schemes